GDSC AI Week
2nd Day

고려대학교 정보대학
Korea University
College of Informatics

전병우
ipcs@korea.ac.kr

What is Machine Learning?

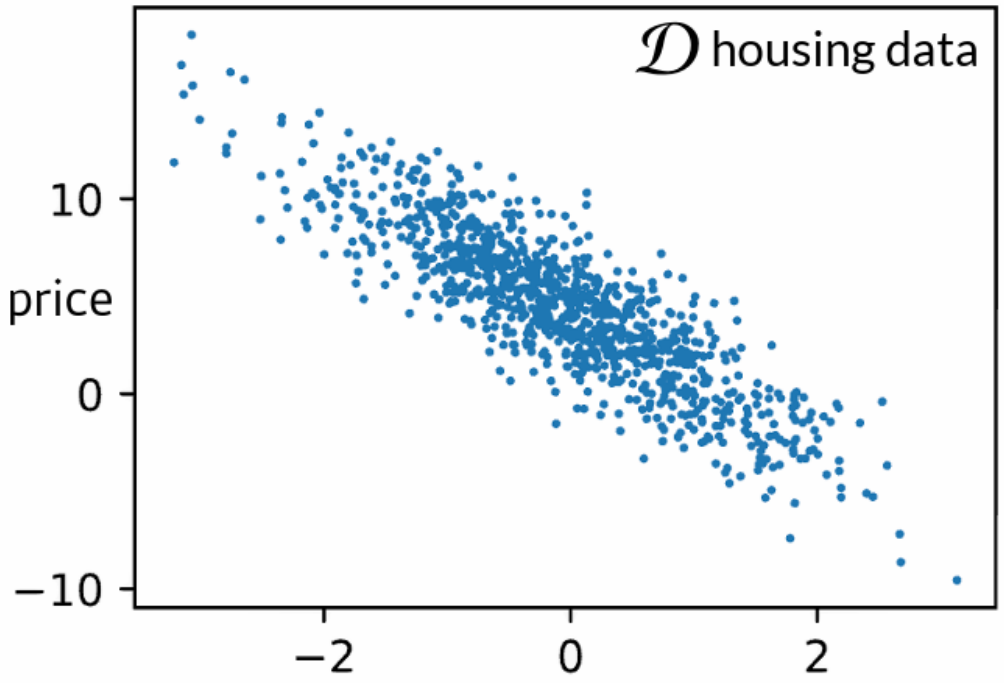
- Mitchell (1997)
  - A computer program is said to learn from experience $\mathcal{E}$ with respect to some class of tasks $\mathcal{T}$ and performance measure $\mathcal{P}$, if its performance at tasks in $\mathcal{T}$, as measure by $\mathcal{P}$, improves with experience $\mathcal{E}$
- Task $\mathcal{T}$ → classification, regression, machine translation, robot control, ⋯
- Performance measure $\mathcal{P}$ → accuracy, error, perplexity, log-likelihood, ⋯
- Experience $\mathcal{E}$ → supervised, unsupervised, reinforcement learning, ⋯

Three ingredients of ML Algorithms

- 데이터 $\mathcal{D}$ → 이미지, 그래프, 텍스트, 비디오, ⋯
  - 차원
  - 데이터 구조
- 모델 $\mathcal{M}$ → 선형 모델, SVM, 의사결정나무(decision tree), 신경망(neural networks), ⋯
  - 입력/출력
  - 파라미터로 구성된 함수
- 목적함수의 학습 $\mathcal{L}$ → L1-loss, L2-loss, Cross-Entropy, ⋯
  - 최적점(Optimal point)은 원하는 성능과 일치해야 한다.

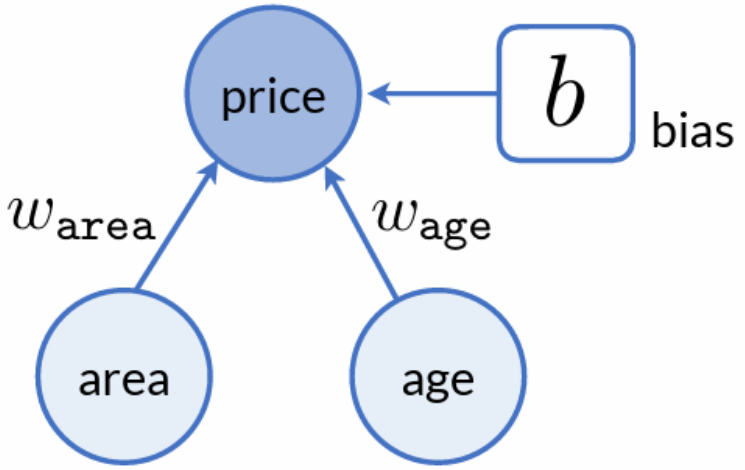
Example: Regression

- $\mathcal{T}$: regression
- $\mathcal{P}$: mean-squared error
- $\mathcal{E}$: supervised learning

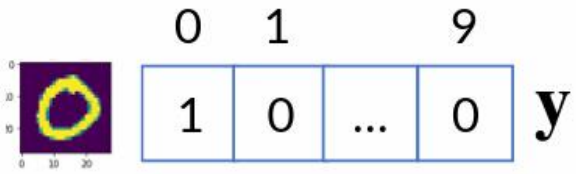


어떻게 데이터 $\mathcal{D}$에 맞는 커브를 찾을 수 있을까?

$$\text{price} = w_{\text{area}} \cdot \text{area} + w_{\text{age}} \cdot \text{age} + b.$$



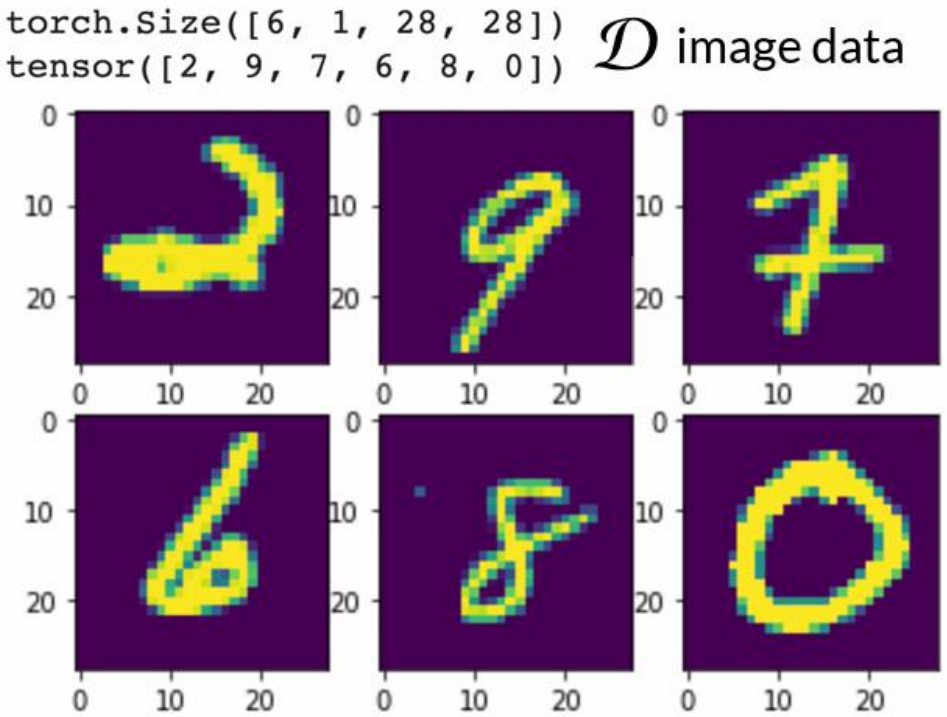
Example: Image Classification



Label: one-hot vector

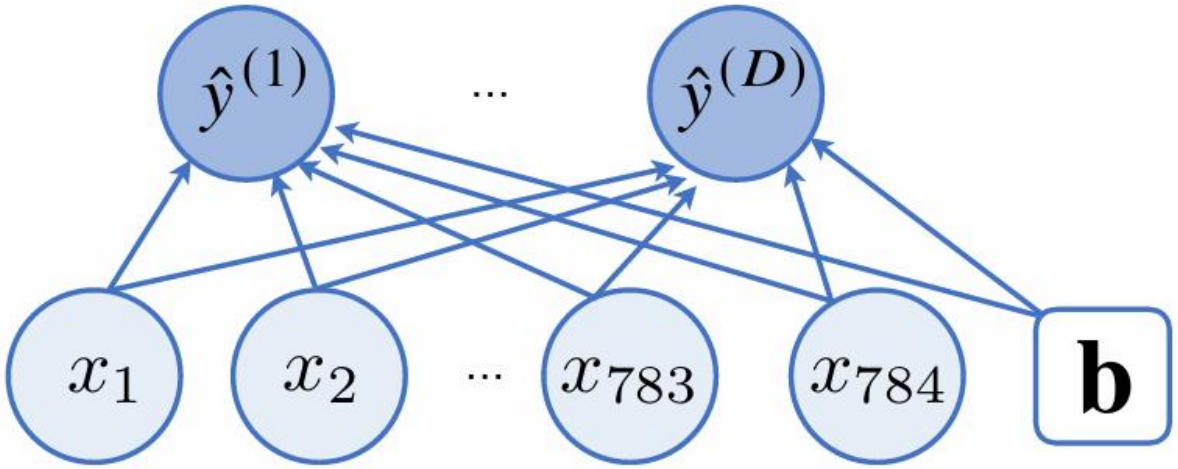
- $\mathcal{T}$: image classification
- $\mathcal{P}$: accuracy
- $\mathcal{E}$: supervised learning

어떻게 분류 목적에 맞는 목적함수를 짤 수 있을까?



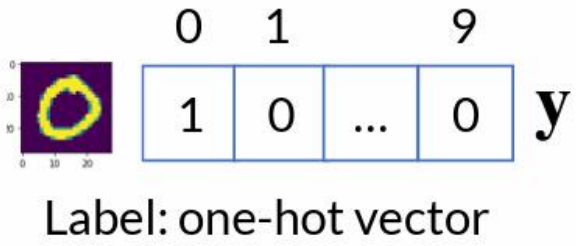
$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{x} + \mathbf{b} \quad \text{or} \quad \hat{y}^{(d)} = \sum_{p \in \{\text{pixel}\}} W_{dp} x_p + b_d \quad d \in \{1, \ldots, D\}$$

weight bias



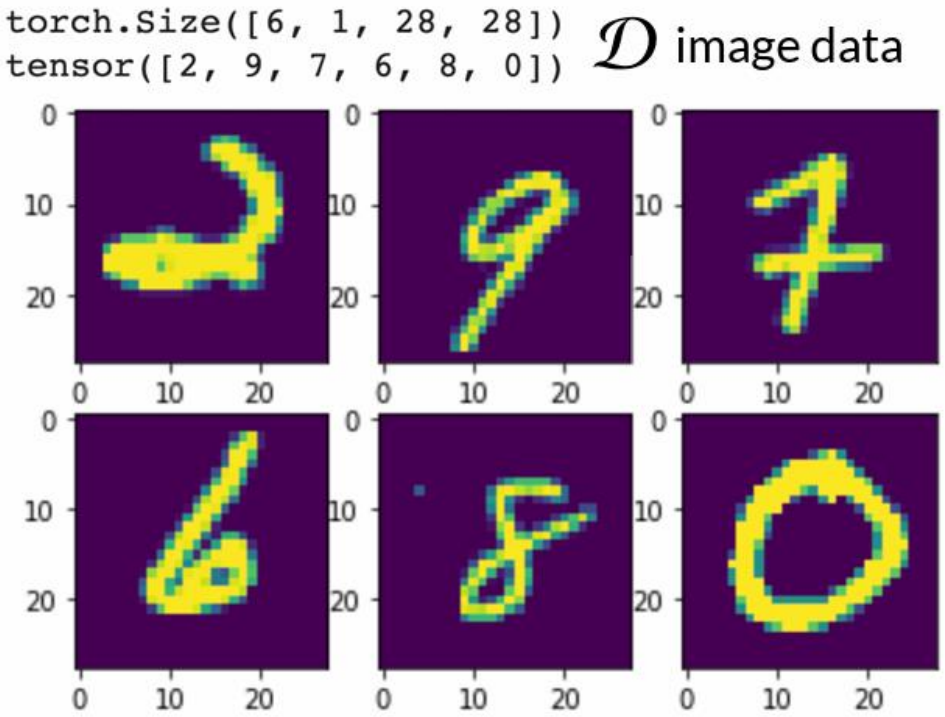
$\mathcal{M}$ linear function

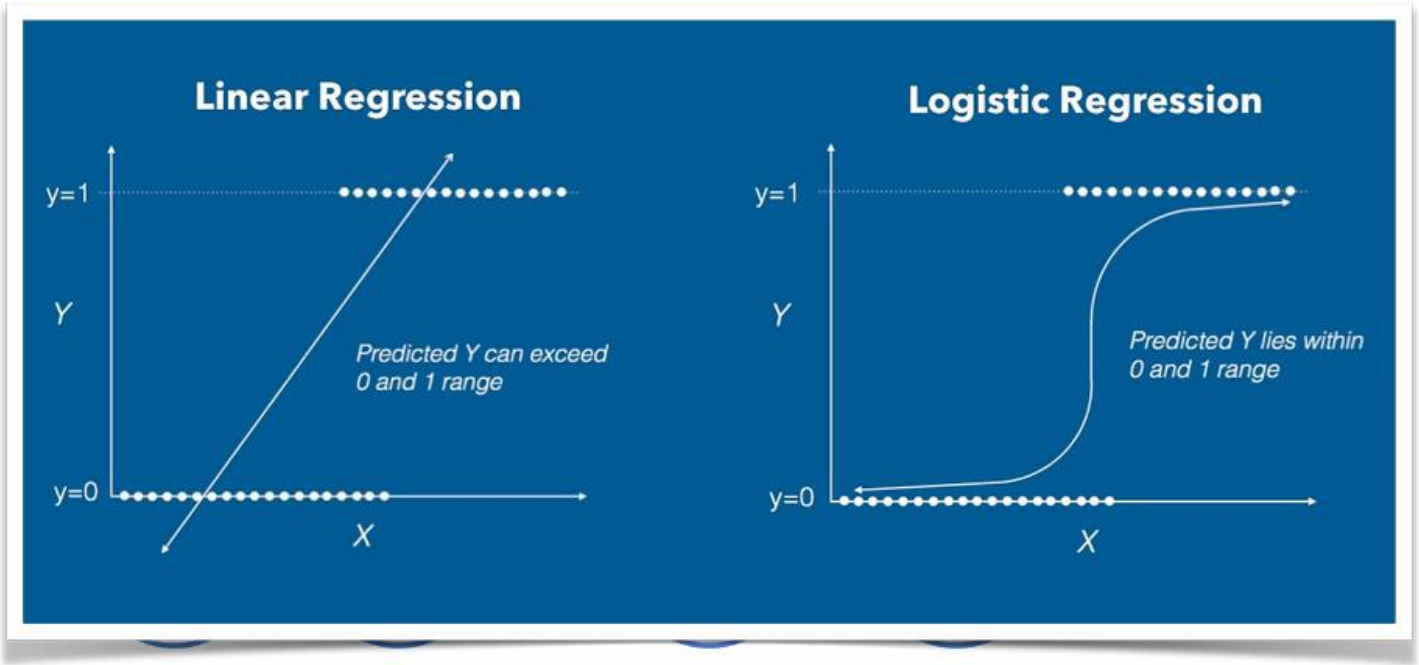
Example: Image Classification



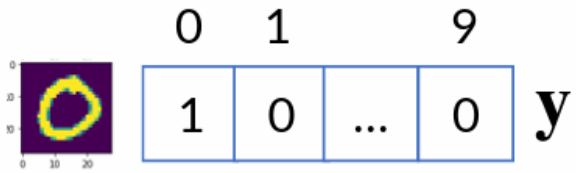
- $\mathcal{T}$: image classification
- $\mathcal{P}$: accuracy
- $\mathcal{E}$: supervised learning

우리는 output vector의 범위를 맞춰줘야 한다!





Example: Image Classification



Label: one-hot vector

- $\mathcal{T}$: image classification
- $\mathcal{P}$: accuracy
- $\mathcal{E}$: supervised learning

여기서 MSE loss를 사용할 수 있을까?

$$\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{y}_i-\hat{\mathbf{y}}_i\|_2^2=\frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D}|y_i^{(d)}-\hat{y}_i^{(d)}|^2$$

$$\hat{\mathbf{y}}=\mathtt{softmax}(\mathbf{W}\mathbf{x}+\mathbf{b})$$

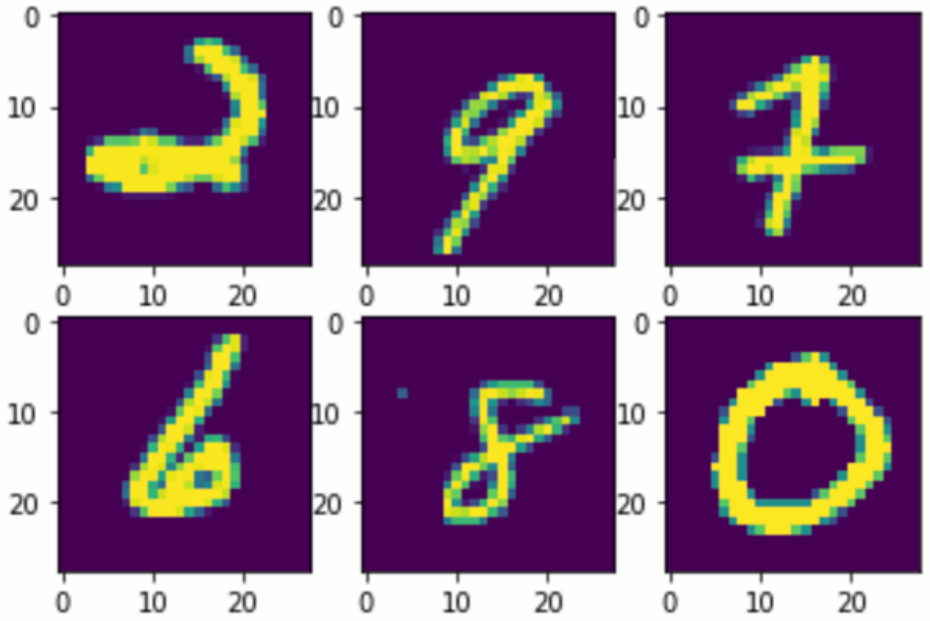
weight bias

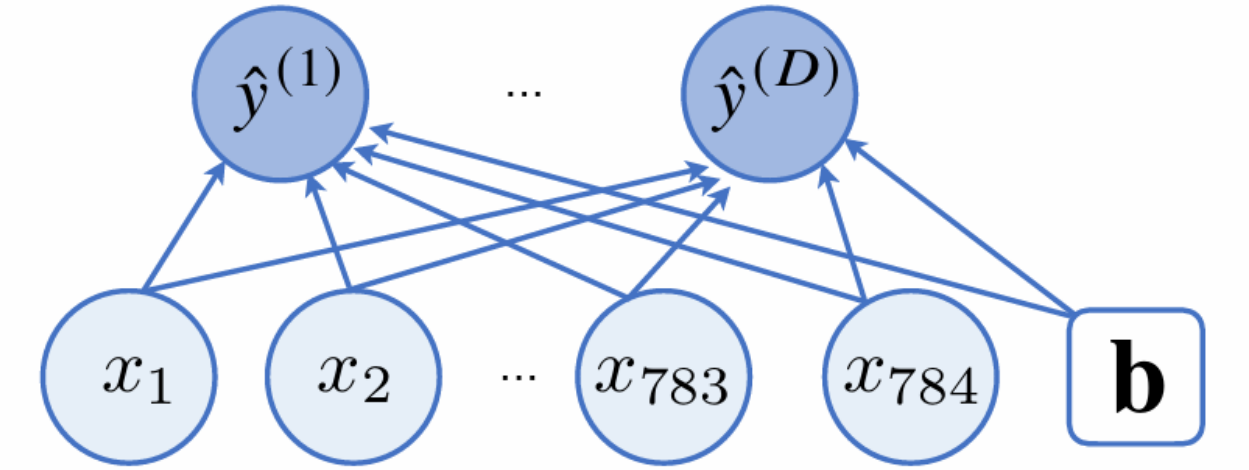
$$\hat{y}^{(d)}=\frac{e^{(\mathbf{W}\mathbf{x}+\mathbf{b})_d}}{\sum_{k=1}^{D}e^{(\mathbf{W}\mathbf{x}+\mathbf{b})_k}}$$

softmax

```
torch.Size([6, 1, 28, 28])
tensor([2, 9, 7, 6, 8, 0])
```

$\mathcal{D}$ image data

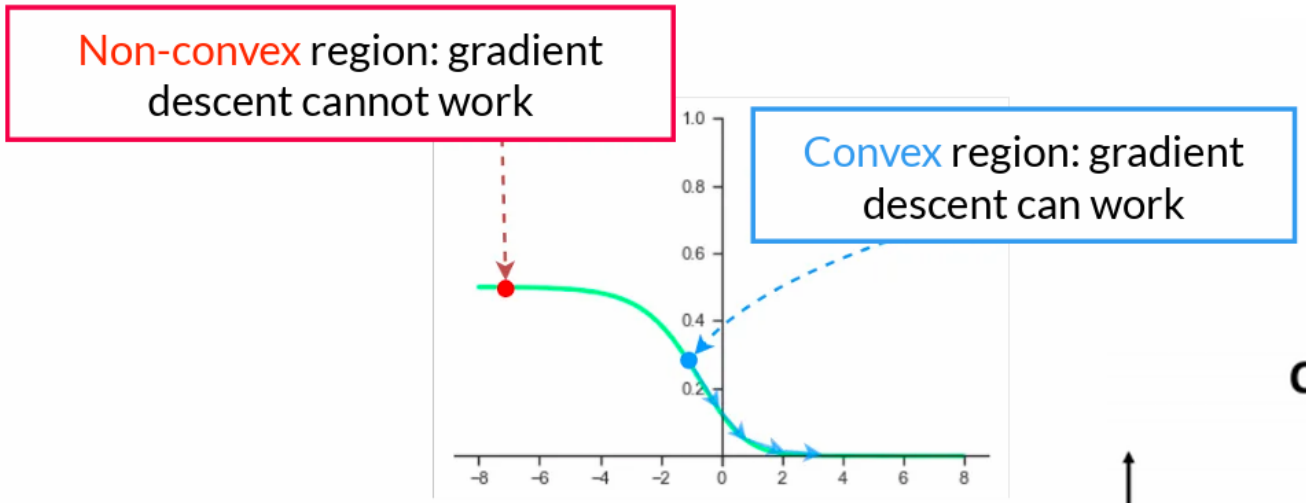


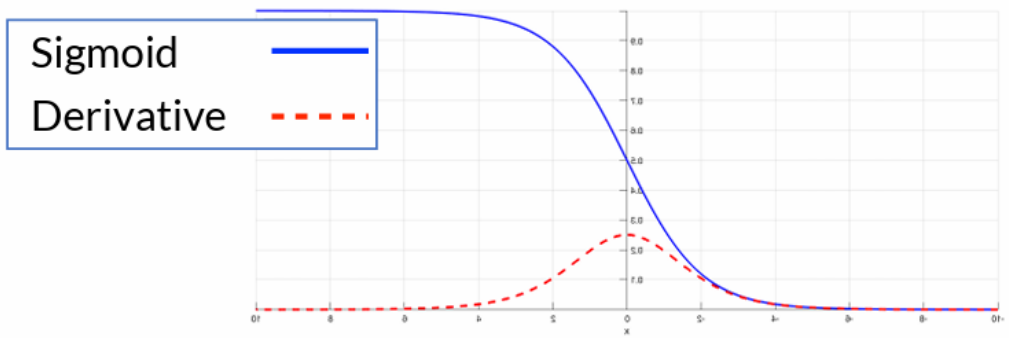


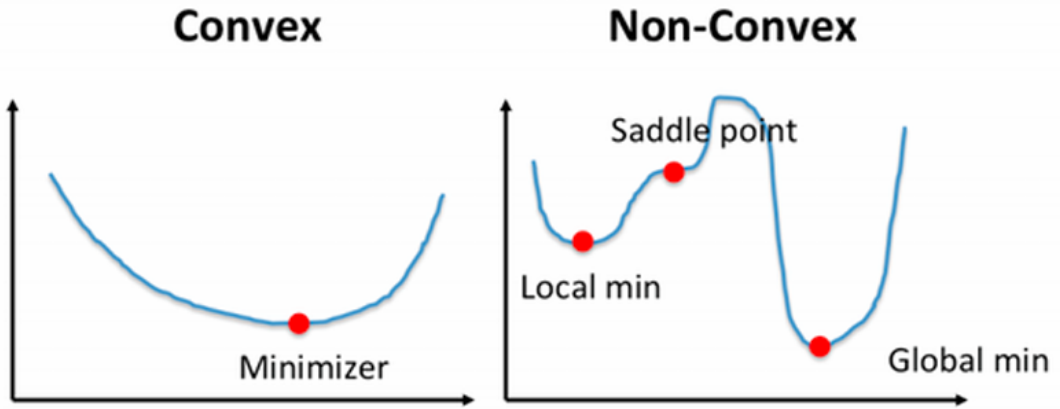
$\mathcal{M}$ multinomial logistic regression (linear neural network)

MSE Loss suffers non-convex optimization in LR!

- 로지스틱 회귀에서 MSE를 사용하면 non-convex한 loss surface가 형성된다.
  - 로지스틱 회귀는 0~1 사이의 값을 가져야 하므로 sigmoid 함수를 사용하기 때문이다.







Remedy: Cross-Entropy Loss

$$\mathbf{y} = (y^{(1)}, \ldots, y^{(D)}) \qquad \hat{\mathbf{y}} = \texttt{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b}) = (\hat{y}^{(1)}, \ldots, \hat{y}^{(D)})$$

1 or 0 (for $y^{(1)}$, $y^{(D)}$) — label: one-hot vector

$\in (0, 1)$ (for $\hat{y}^{(1)}$, $\hat{y}^{(D)}$) — model output

PMF of Categorial Distribution

$$\mathbb{P}_{\mathcal{M}}(Y = \mathbf{y}|\mathbf{x}) = \left(\hat{y}^{(1)}\right)^{y^{(1)}} \times \cdots \times \left(\hat{y}^{(D)}\right)^{y^{(D)}} = \hat{y}^{(*)}$$

when $y^{(*)} = 1$

$\mathcal{M}$: model

Y: random variable

$\mathbf{x}$: input

따라서 softmax의 결과는 조건부 확률로 해석할 수 있습니다

Remedy: Cross-Entropy Loss

$$\mathbf{y} = (y^{(1)}, \ldots, y^{(D)}) \qquad \hat{\mathbf{y}} = \texttt{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b}) = (\hat{y}^{(1)}, \ldots, \hat{y}^{(D)})$$

1 or 0 (for $y^{(1)}$, $y^{(D)}$); $\in (0, 1)$ (for $\hat{y}^{(1)}$, $\hat{y}^{(D)}$)

label: one-hot vector

model output

PMF of Categorial Distribution

$$\log \mathbb{P}_{\mathcal{M}}(Y = \mathbf{y}|\mathbf{x}) = \sum_{d=1}^{D} y^{(d)} \log \hat{y}^{(d)}$$

여기서 로그 함수를 취할 수 있다!

Remedy: Cross-Entropy Loss

$$\mathbf{y} = (y^{(1)}, \ldots, y^{(D)}) \qquad \hat{\mathbf{y}} = \mathtt{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b}) = (\hat{y}^{(1)}, \ldots, \hat{y}^{(D)})$$

$y^{(d)}$: 1 or 0 — label: one-hot vector

$\hat{y}^{(d)} \in (0, 1)$ — model output

PMF of Categorial Distribution

$$\log \mathbb{P}_{\mathcal{M}}(Y = \mathbf{y}|\mathbf{x}) = \sum_{d=1}^{D} y^{(d)} \log \hat{y}^{(d)}$$

이를 통해 negative log-likelihood를 얻을 수 있습니다

$$\mathbb{E}_{(\mathbf{X},Y)\sim\mathcal{D}} \left[-\log \mathbb{P}_{\mathcal{M}}(Y|\mathbf{X})\right] \overset{\text{M.C}}{\approx} -\frac{1}{N}\sum_{i=1}^{N} \log \mathbb{P}_{\mathcal{M}}(Y = \mathbf{y}_i|\mathbf{x}_i)$$

정보 격차 최소화

$$= -\frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D} y_i^{(d)} \log \hat{y}_i^{(d)} \overset{\text{M.C}}{\approx} H(\mathbf{y}, \hat{\mathbf{y}})$$

Cross-Entropy

Remedy: Cross-Entropy Loss

A linear function:

$f(x) = ax - b$

$a = 3$

$b = 2$

The squared error loss of the sigmoid function if the label is 0:

$y = \left(\frac{1}{1+e^{-f(x)}}\right)^2$

The squared error loss of the sigmoid function if the label is 1:

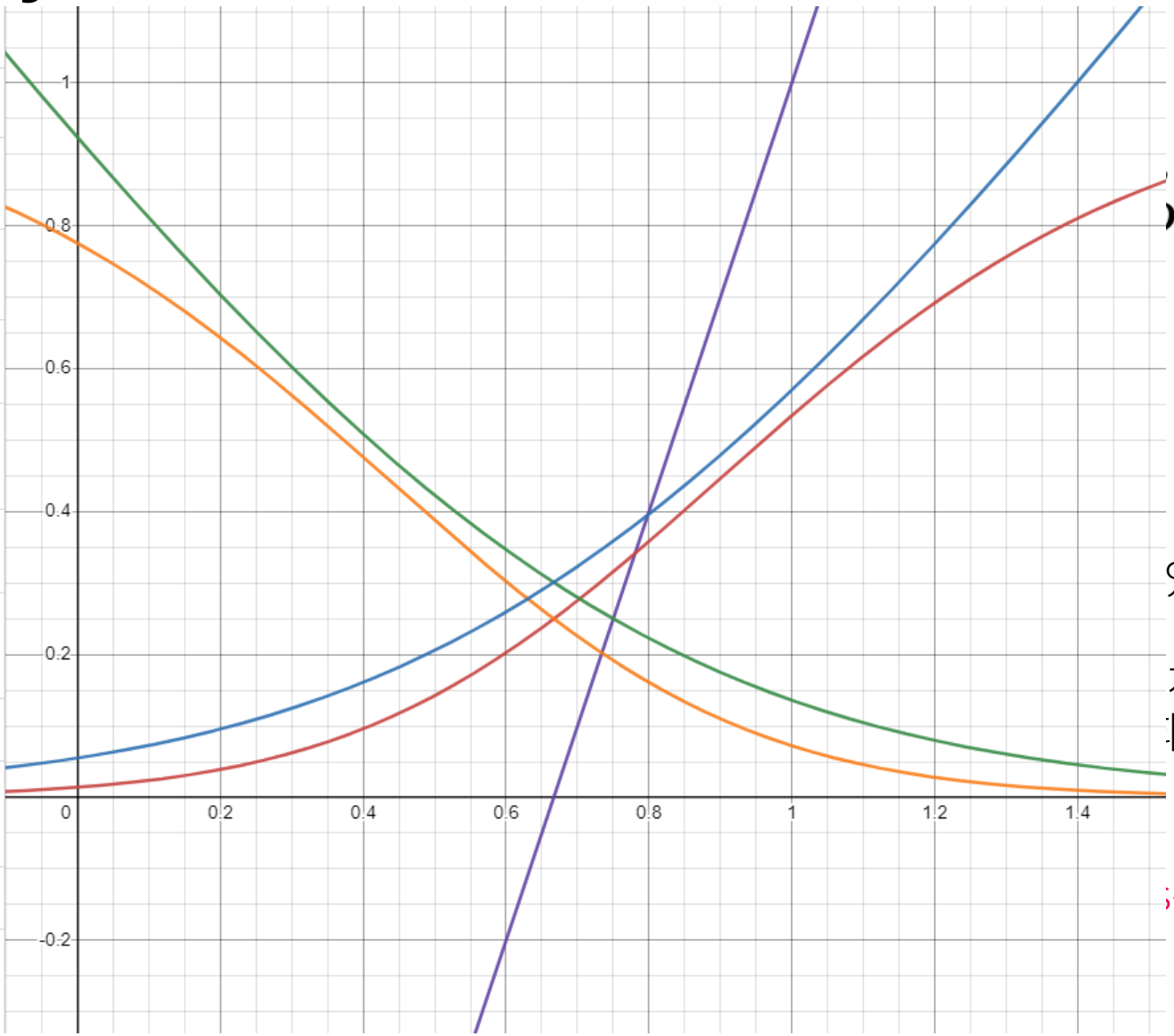
$y = \left(1 - \frac{1}{1+e^{-f(x)}}\right)^2$

The log loss of the sigmoid function if the label is 0:

$y = -\log\left(\frac{1}{1+e^{-f(x)}}\right)$

The log loss of the sigmoid function if the label is 1:

$y = -\log\left(1 - \frac{1}{1+e^{-f(x)}}\right)$



1)

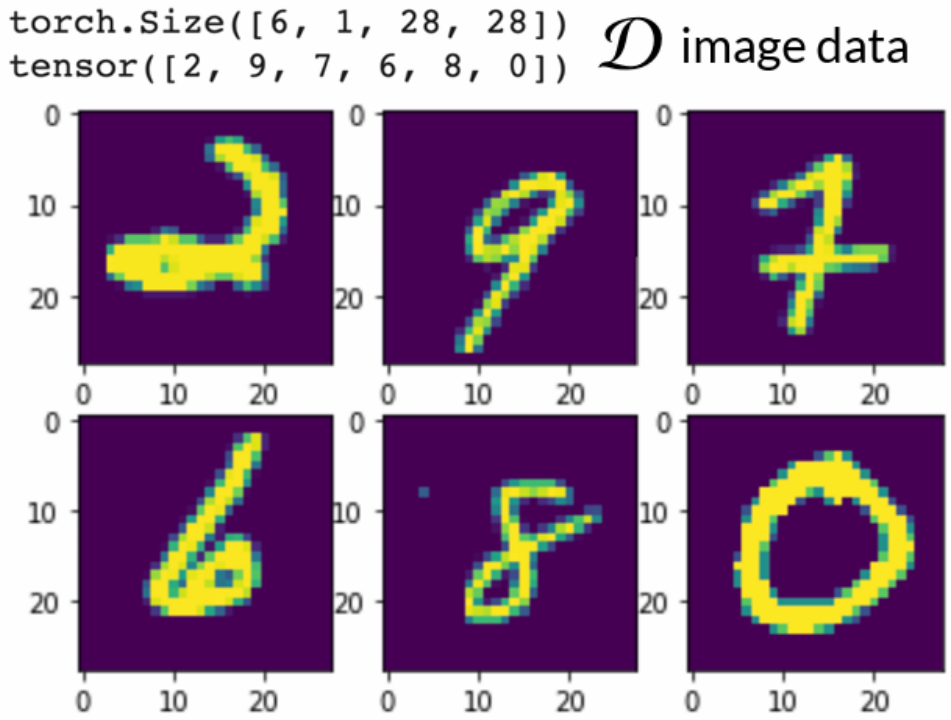
있습니다

차

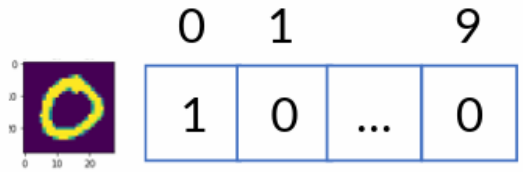
-Entropy

Example: Image Classification

- $\mathcal{T}$: image classification
- $\mathcal{P}$: accuracy
- $\mathcal{E}$: supervised learning



$\mathcal{D}$ image data



Label: one-hot vector

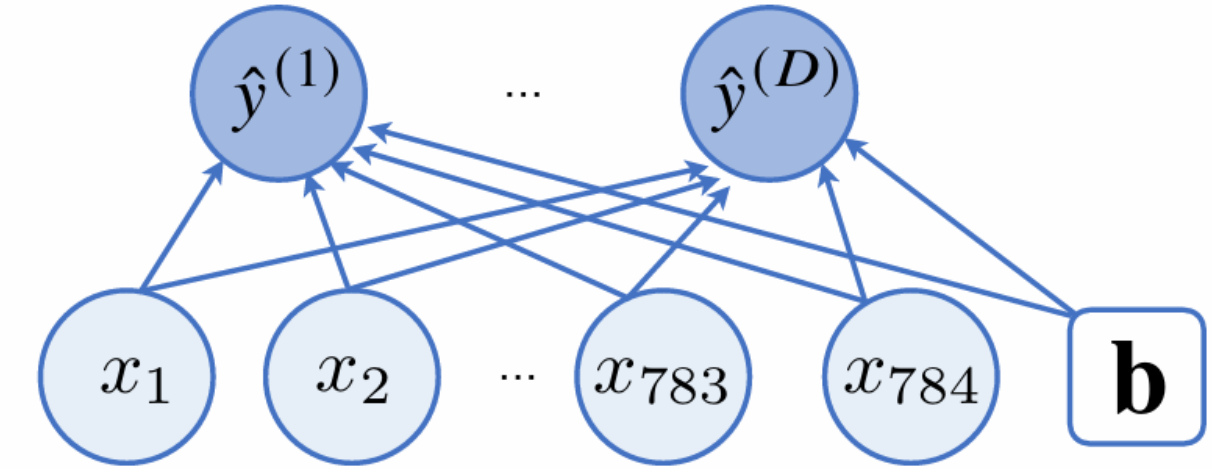
$\mathcal{L}$ Cross-Entropy Loss $-\frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D} y_i^{(d)} \log \hat{y}_i^{(d)}$

$$\hat{\mathbf{y}} = \mathtt{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

weight bias

$$\hat{y}^{(d)} = \frac{e^{(\mathbf{W}\mathbf{x}+\mathbf{b})_d}}{\sum_{k=1}^{D} e^{(\mathbf{W}\mathbf{x}+\mathbf{b})_k}}$$

softmax

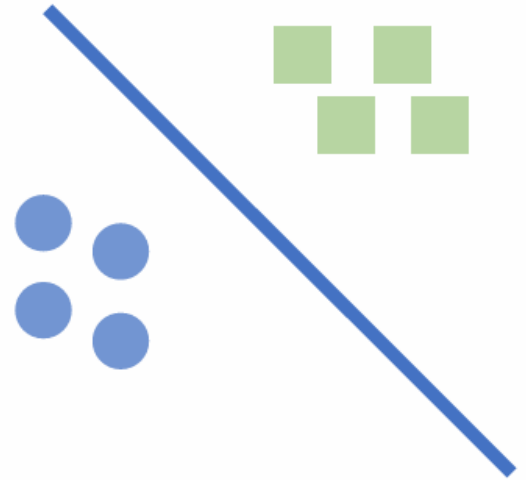


$\mathcal{M}$ multinomial logistic regression (linear neural network)

To Nonlinear, and Beyond!

Separating Hyperplane Problem

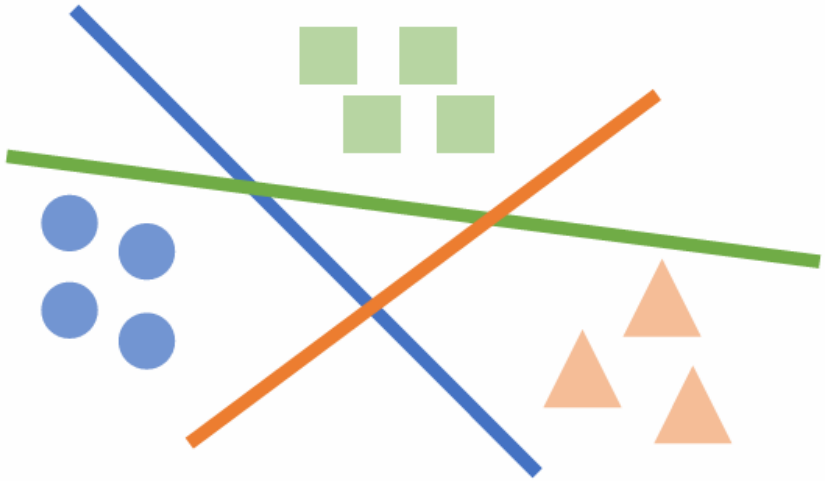
- 이진 분류 문제는 결국 두 class에 대한 초평면(hyperplane)을 찾는 것과 같다.



Separating Hyperplane Problem

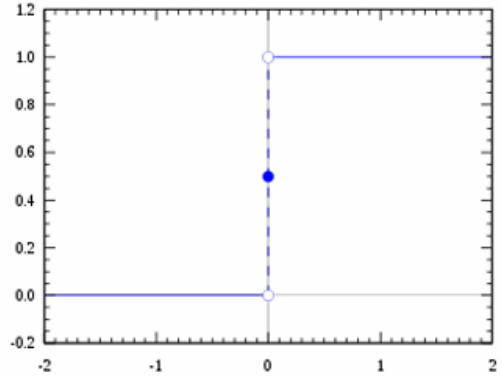
- 이진 분류 문제는 결국 두 class에 대한 초평면(hyperplane)을 찾는 것과 같다.
- One-hot vector는 multinomial classification을 보여준다.

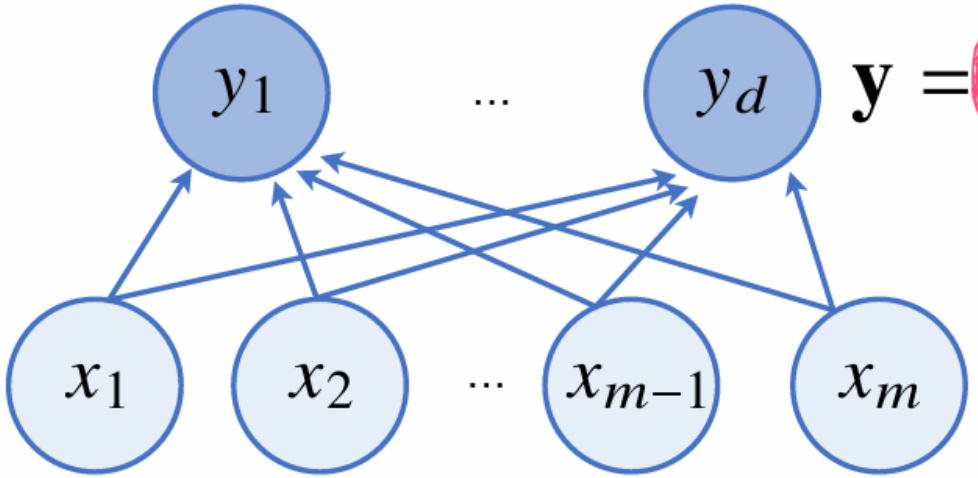
즉, multinomial classification을 위해서는
multiple hyperplanes가 필요함을 뜻합니다.



Perceptrons (1957)

- 만약 σ가 단위 계단 함수(Heaviside function)라면 이를 퍼셉트론이라 한다.

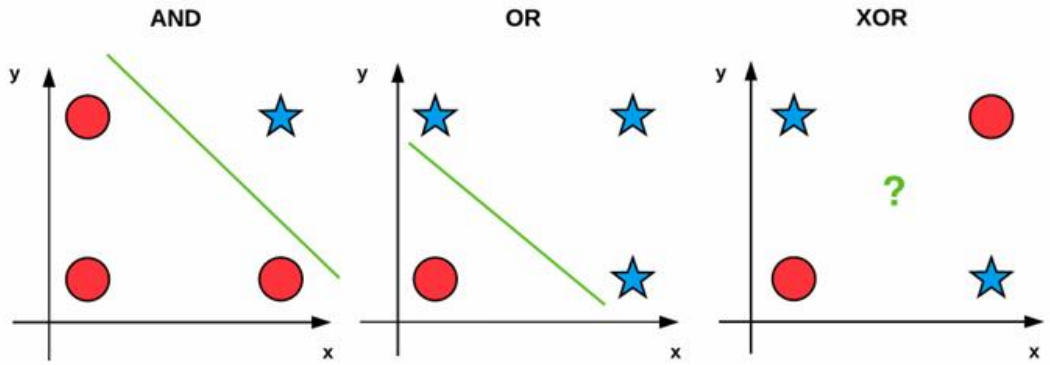


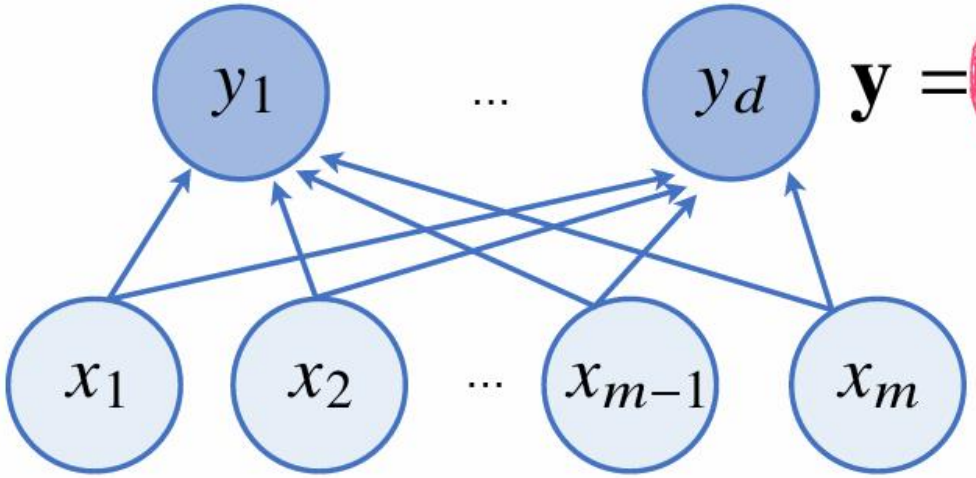


$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \begin{cases} 1 & : \mathbf{W}\mathbf{x} + \mathbf{b} \geq 0 \\ 0 & : \text{otherwise} \end{cases}$$

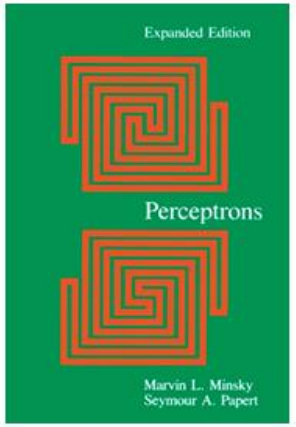
Perceptrons (1957)

- 그러나 퍼셉트론은 XOR 문제 등 non-separable 문제를 풀 수 없다.





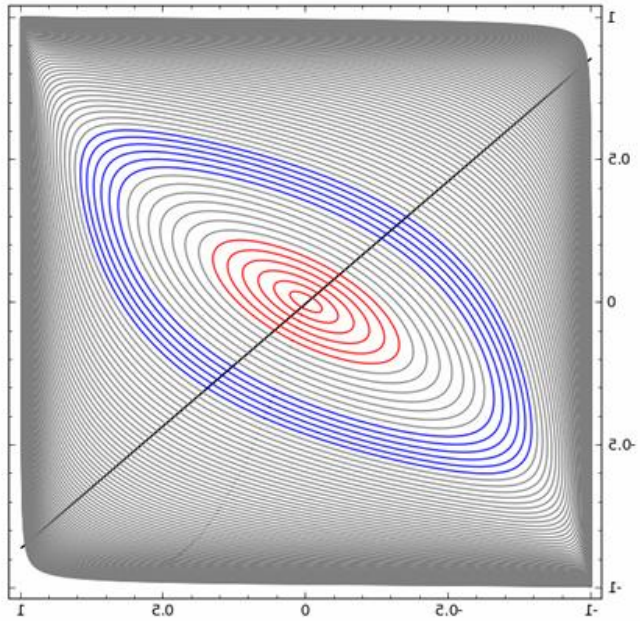
$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \begin{cases} 1 & : \mathbf{W}\mathbf{x} + \mathbf{b} \geq 0 \\ 0 & : \text{otherwise} \end{cases}$$



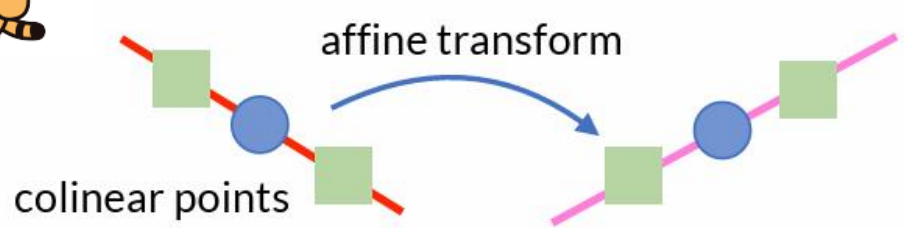
Rosenblatt vs Minsky

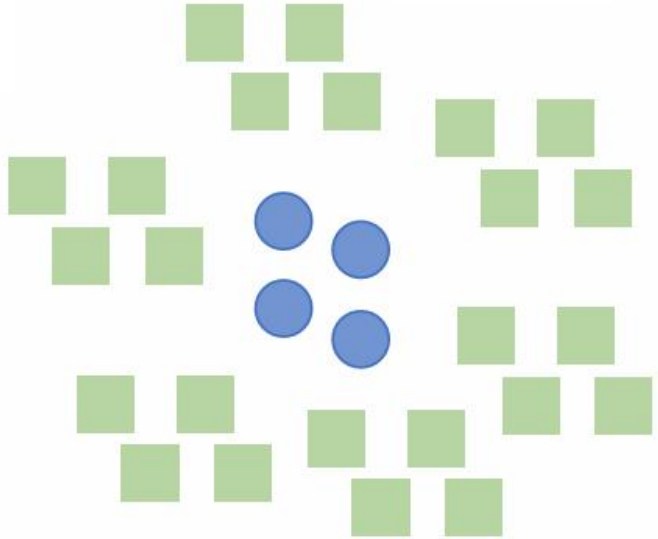
Beyond Linear Models

- 만약 경계(decision boundary)가 nonlinear라면, 우리는 features를 transform해 non-separable space에서 separable space로 mapping해야 한다.
  - 그러나 선형 모델은 이를 할 수 없다!



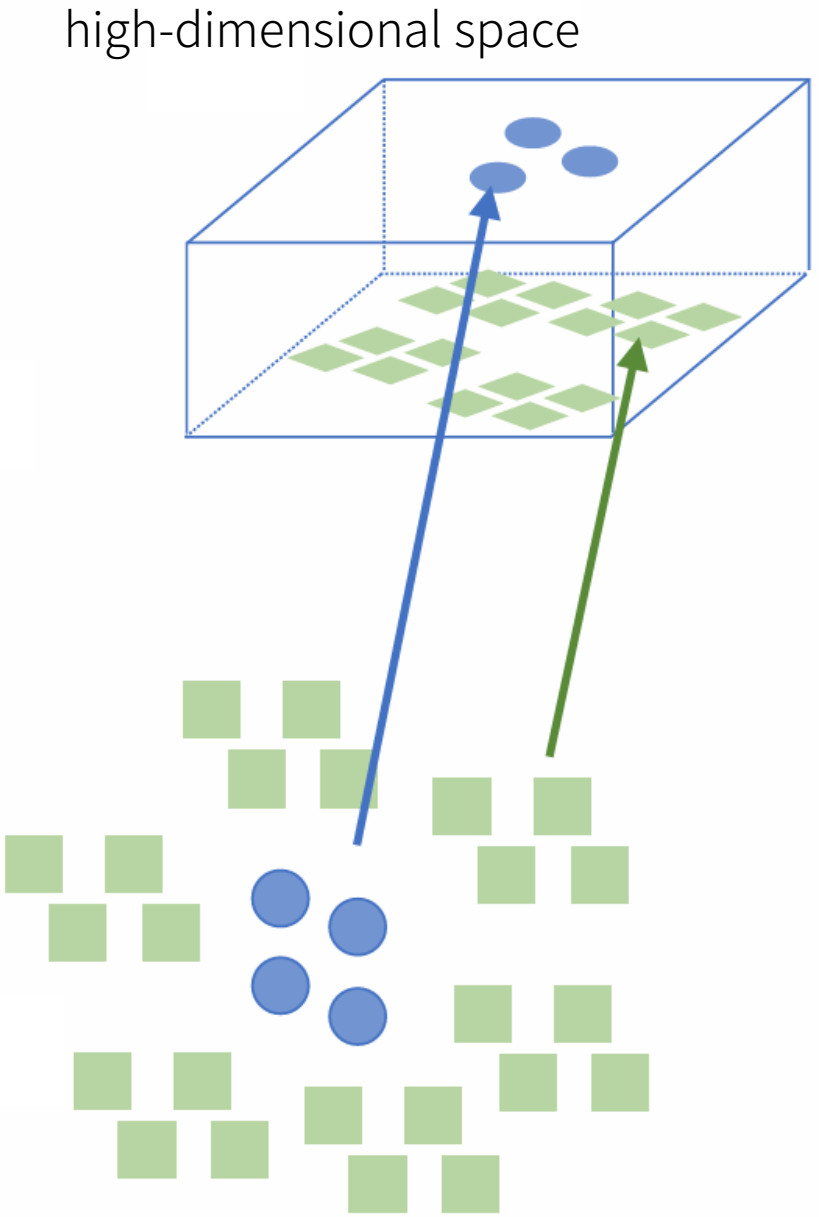
또 나다. 왜지?





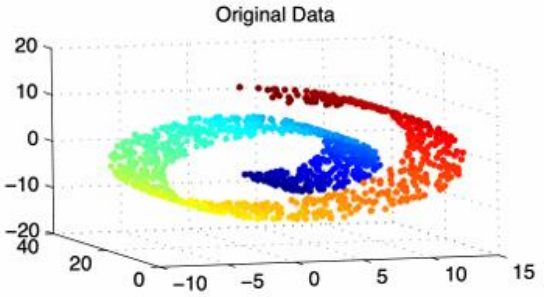
Beyond Linear Models

- 만약 경계(decision boundary)가 nonlinear라면, 우리는 features를 transform해 non-separable space에서 separable space로 mapping해야 한다.
  - 그러나 선형 모델은 이를 할 수 없다!
- 우리는 데이터를 smooth representation을 이용해 high-dimensional space으로 mapping할 수 있다.
  - 이것은 다양체 가설(manifold hypothesis)가 만족한다면 가능하다.



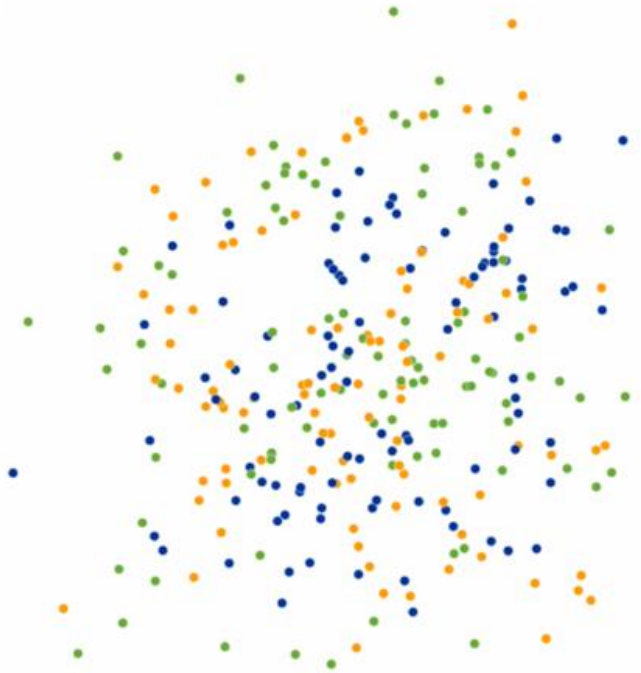
Manifold Hypothesis

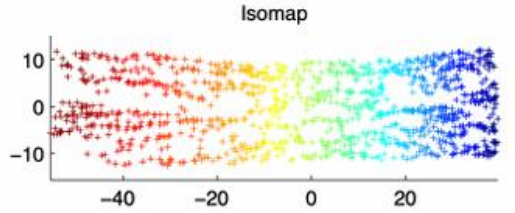
이건 데이터에 대한 가설이지
모델에 대한 것이 아니다!



- **Manifold Learning**

Problem: Given points $x_1, \ldots, x_n \in \mathbb{R}^D$ that lie on a d-dimensional manifold M that can be described by a single coordinate chart $f : M \to \mathbb{R}^d$, find $y_1, \ldots, y_n \in \mathbb{R}^d$, where $y_i \stackrel{\text{def}}{=} f(x_i)$.

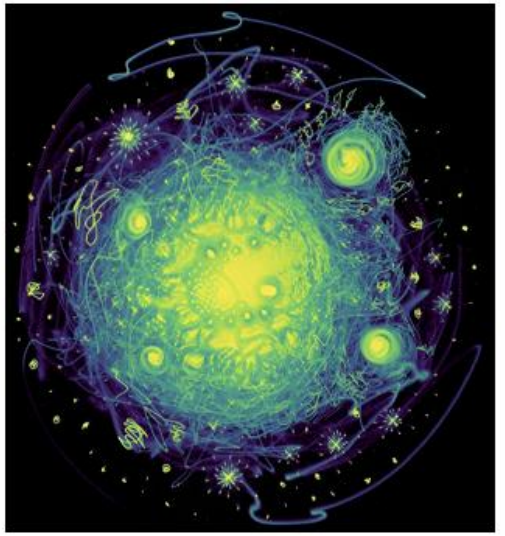




- **Manifold hypothesis**

Hypothesis: high dimensional data tend to lie in the vicinity of a low dimensional manifold

C.Fefferman



Integers colored by prime divisibility

Testing the Manifold Hypothesis, Fefferman et al., J. Amer. Math. Soc. 2016

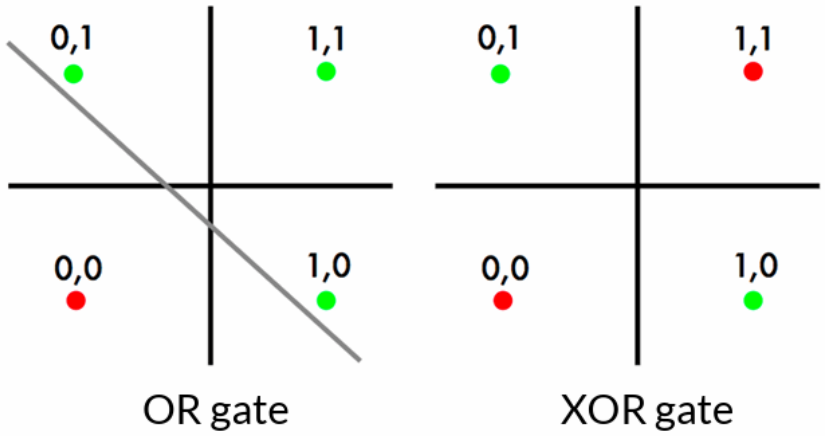
앞으로 Manifold Hypothesis를 가정하자!

(Recap) Percetron

- 퍼셉트론은 F.Rosenblatt에 의해 제안되었다.
- XOR 문제

F.Rosenblatt

	OR	XOR
(True, True)	1	0
(True, False)	1	1
(False, True)	1	1
(False, False)	0	0



어떻게 하면 XOR 문제를 해결하는 초평면을 찾을 수 있을까?

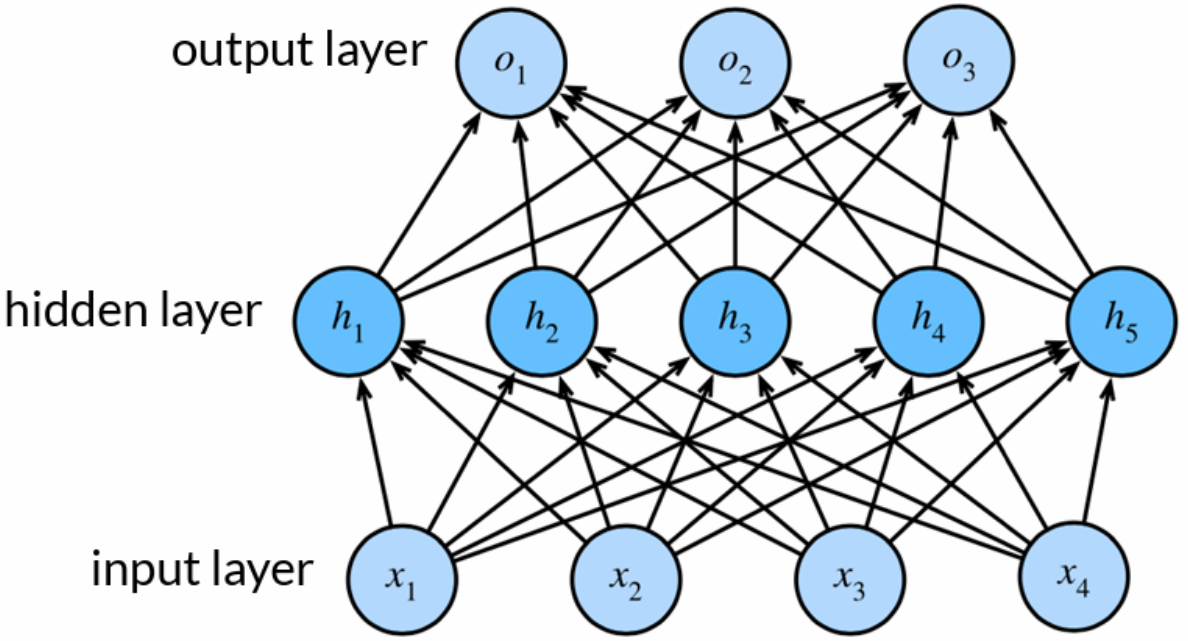
From Linear to Nonlinear

- 우리는 퍼셉트론을 여러 층(layer) 쌓고 nonlinear activation function을 사용함으로써 XOR 문제를 풀 수 있다.
  - nonlinear activation function을 쓰지 않으면 문제를 해결할 수 없다.

- Linear form

$$h_j = \sum_i w_{ij}^{(1)} x_i = \mathbf{w}_j^{(1)} \mathbf{x}$$

$$o_k = \sum_j w_{jk}^{(2)} h_j = \mathbf{w}_k^{(2)} \mathbf{h}$$



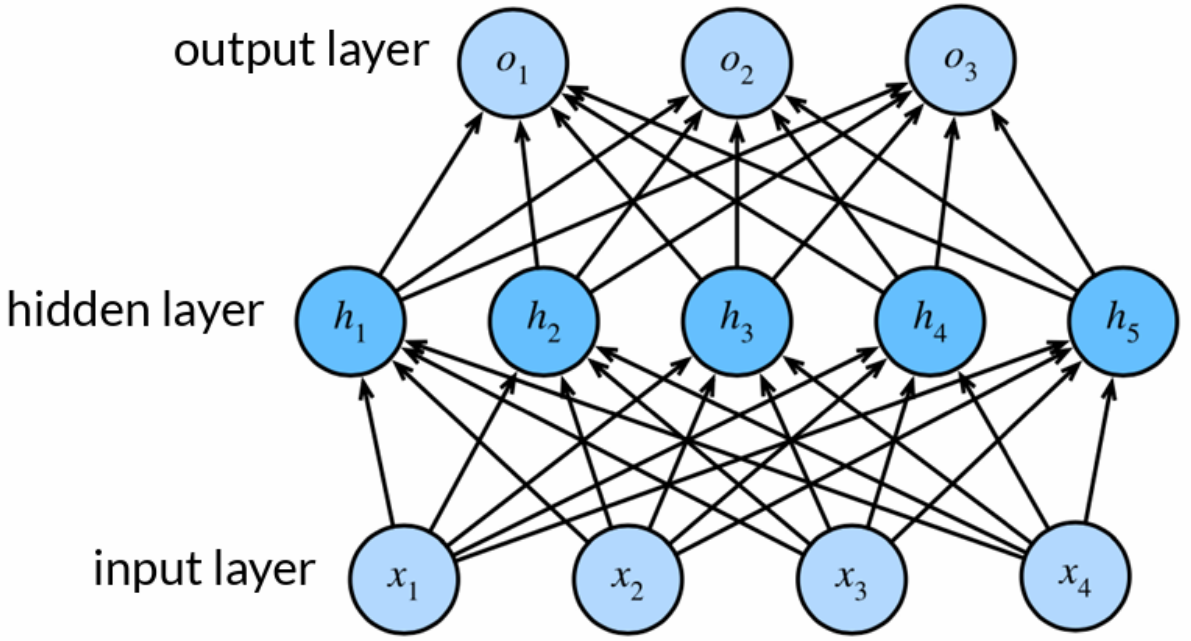
From Linear to Nonlinear

- 우리는 퍼셉트론을 여러 층(layer) 쌓고 nonlinear activation function을 사용함으로써 XOR 문제를 풀 수 있다.
  - nonlinear activation function을 쓰지 않으면 문제를 해결할 수 없다.

- Nonlinear form

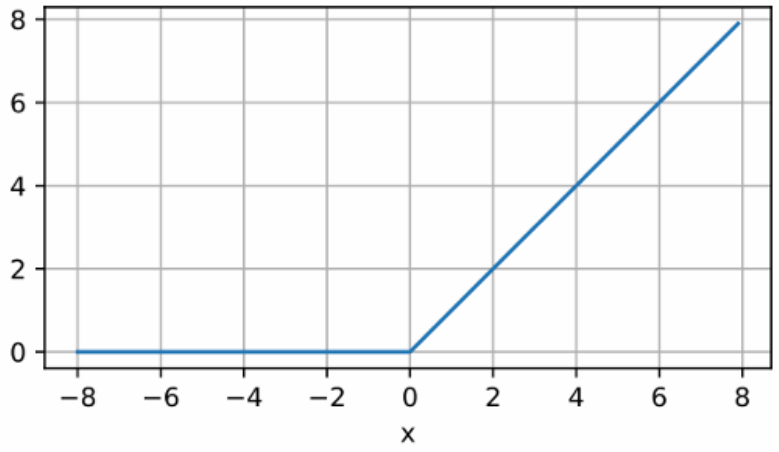
$$h_j = \varphi\left(\sum_i w_{ij}^{(1)} x_i\right) = \varphi(\mathbf{w}_j^{(1)}\mathbf{x})$$

$$o_k = \sum_j w_{jk}^{(2)} h_j = \mathbf{w}_k^{(2)}\mathbf{h}$$

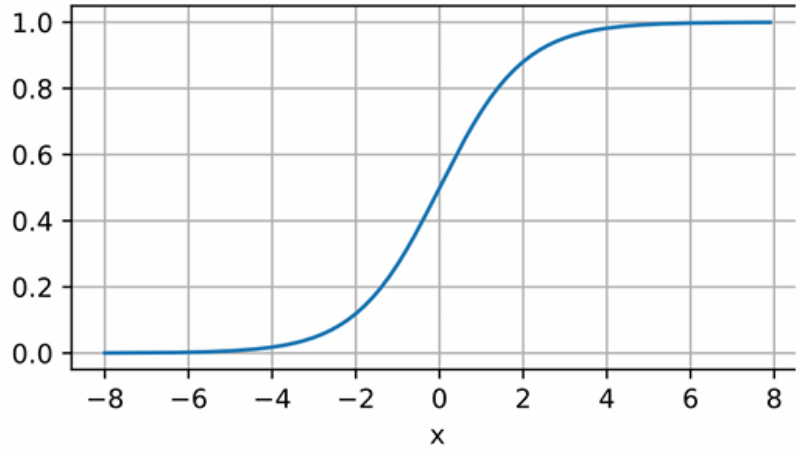


Activation Functions

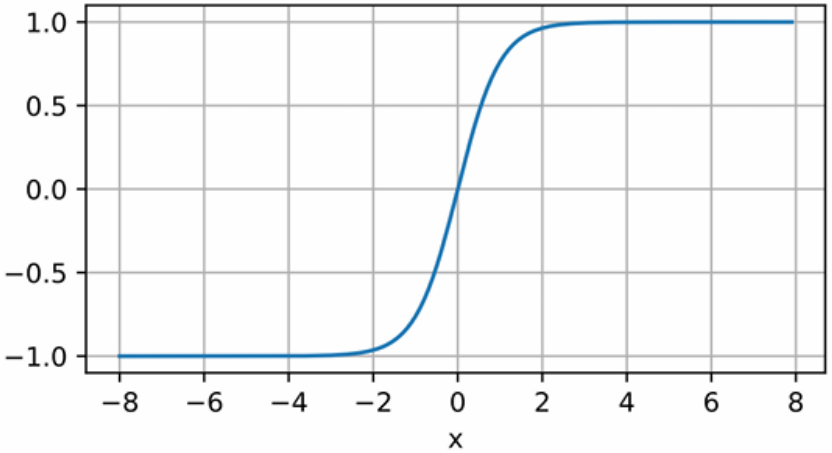
ReLU



Sigmoid



Hyperbolic Tangent



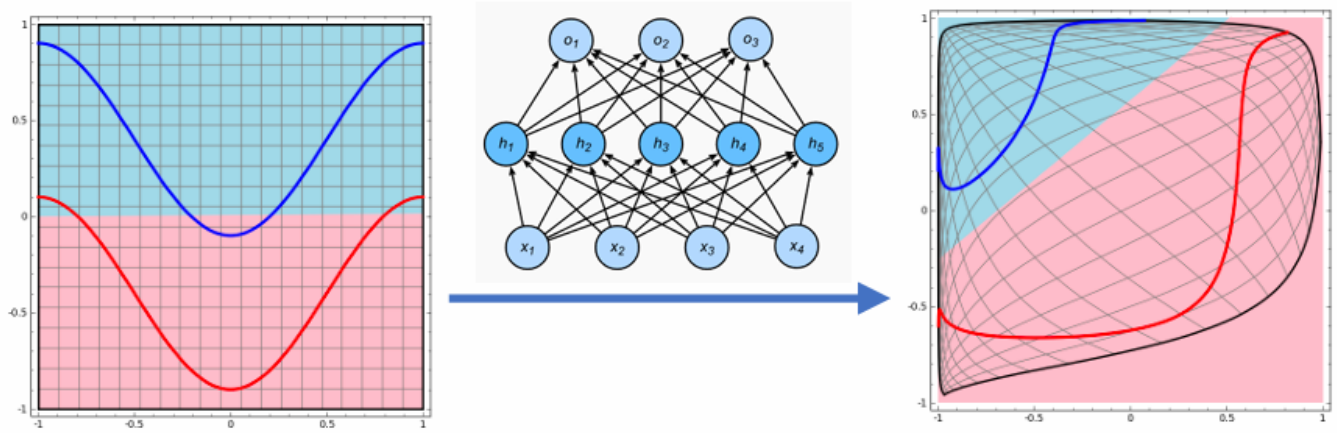
$$\mathrm{ReLU}(x) = \max(x, 0)$$

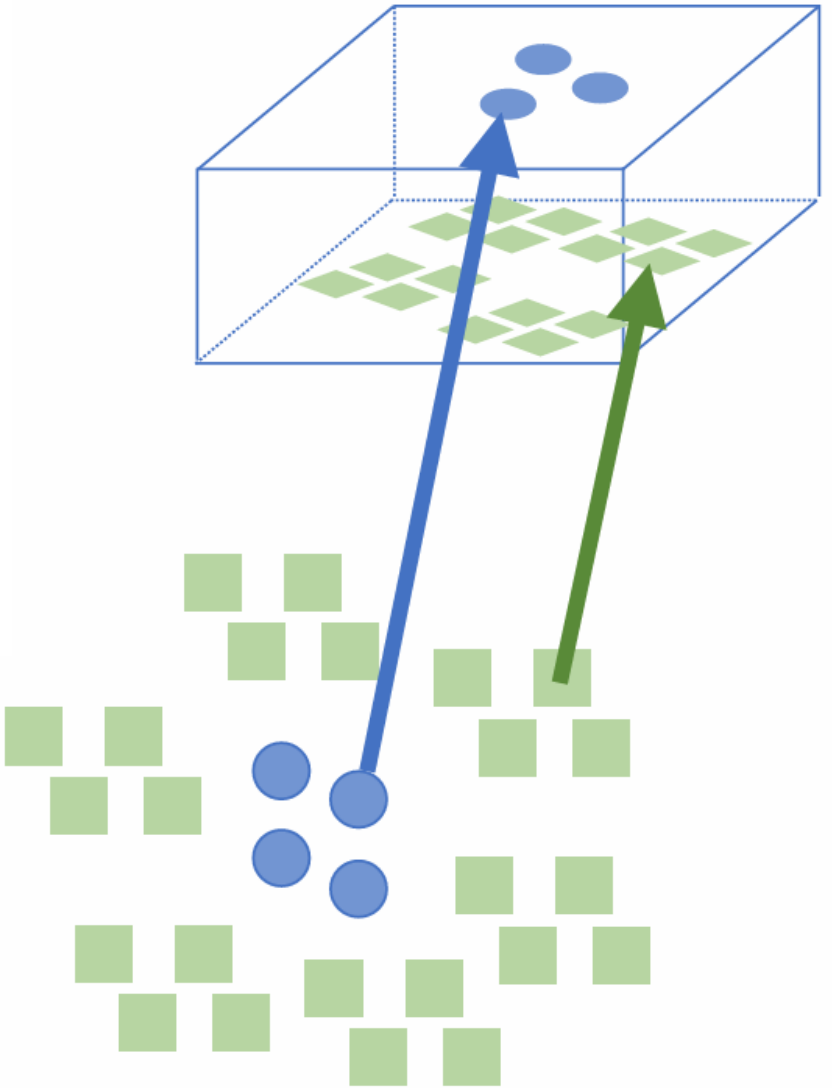
$$\mathrm{Sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$

(Recap) Manifold Learning

- 우리는 데이터를 smooth representation을 이용해 high-dimensional space으로 mapping할 수 있다.
  - 이것은 다양체 가설(manifold hypothesis)가 만족한다면 가능하다.
  - 그 첫 예시로 multi-layer perceptrons을 이용하자.





Theoretical Power of Neural Networks

- Universal Approximation Theorem (Hornik & Cybenko, 1989)

Universal approximation theorem — Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$, so $\sigma \circ x$ denotes σ applied to each component of x.

Then σ is not polynomial if and only if for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m)$, $\varepsilon > 0$ there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ such that

classifier

hidden-layer

$$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon$$

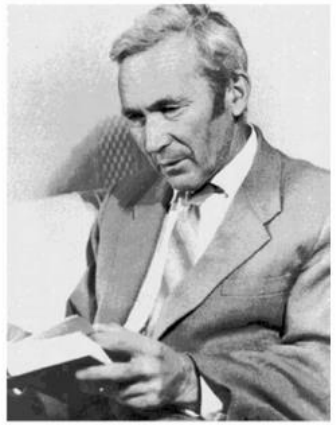
where $g(x) = C \cdot (\sigma \circ (A \cdot x + b))$

이 이론은 neural networks의 존재성을 보장해줄 뿐 찾는 법에 대한 것이 아닙니다.

(Universal Approximation Theorem) For a given arbitrary continuous function on bounded domain and an error bound, there always exists a one-hidden-layer neural network which can approximate the given continuous function within the error bound.

Theoretical Power of Neural Networks

- Universal Approximation Theorem (Hornik & Cybenko, 1989)
- Kolmogorov-Arnold Representation Theorem (Hilbert's 13th problem)



A. Kolmogorov

V. Arnold

D. Sprecher

$$f(\mathbf{x}) = \sum_{q=0}^{2n} \Phi \left(\sum_{p=1}^{n} \lambda_p \phi(x_p + \eta q) + q \right)$$

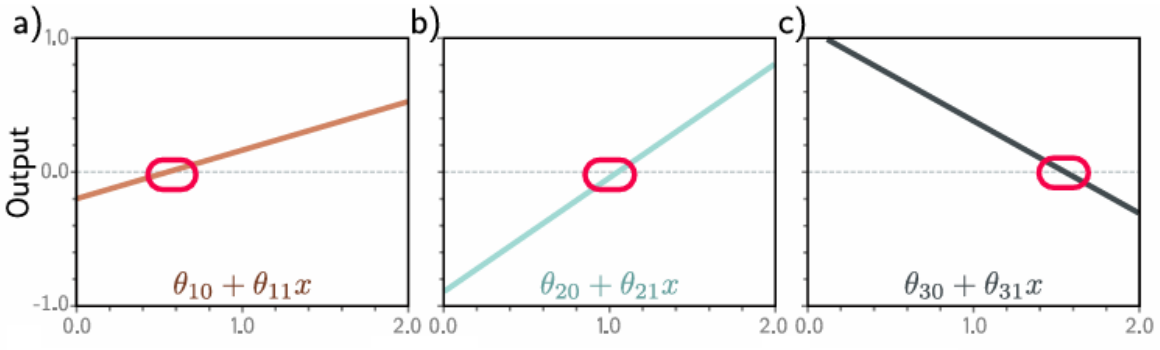
continuous — monotonic increasing

(Superposition Theorem, 1965)

For a given arbitrary continuous function on bounded domain, there always exists the above representation with a monotonic increasing ϕ and a continuous function Φ

On the structure of continuous functions of several variables, D. Sprecher, **Trans. Amer. Math. Soc.**, 1965

How does Neural Network approximate?

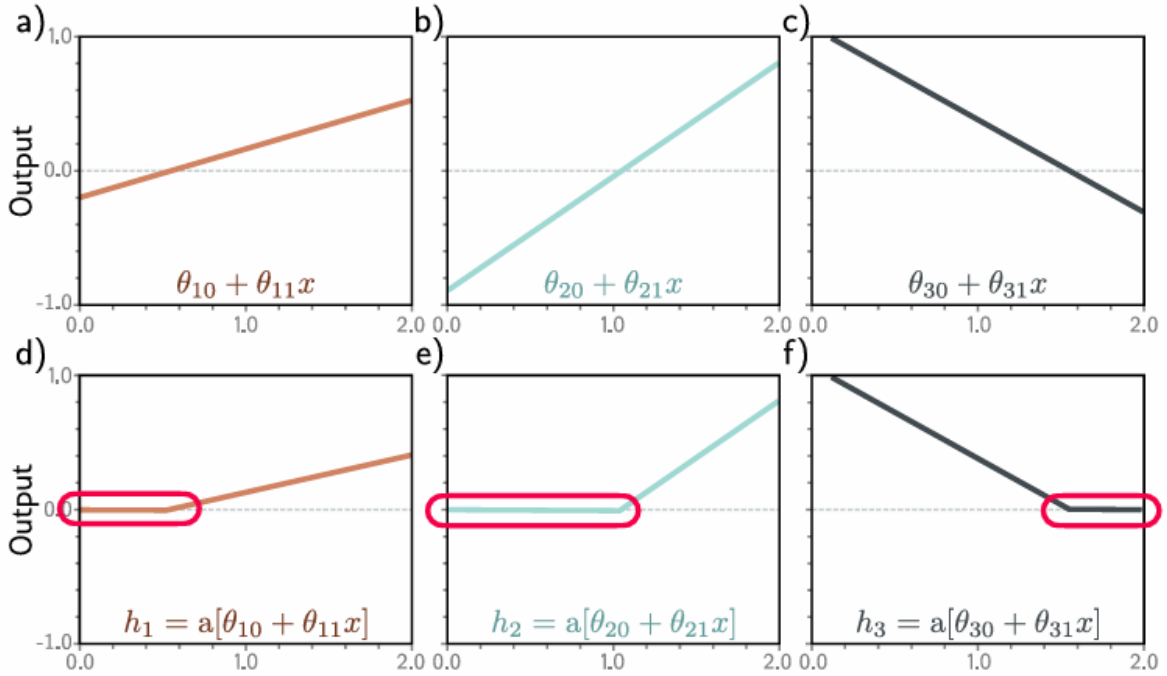


linear transformation in the first layer

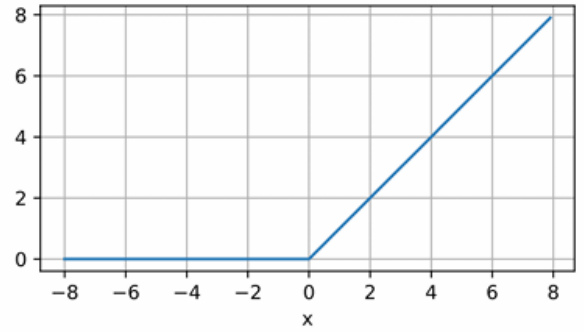
첫 layer에서 fold될 영역이 설정되며 이것이 곧 모델의 expressive power를 결정합니다

.

How does Neural Network approximate?



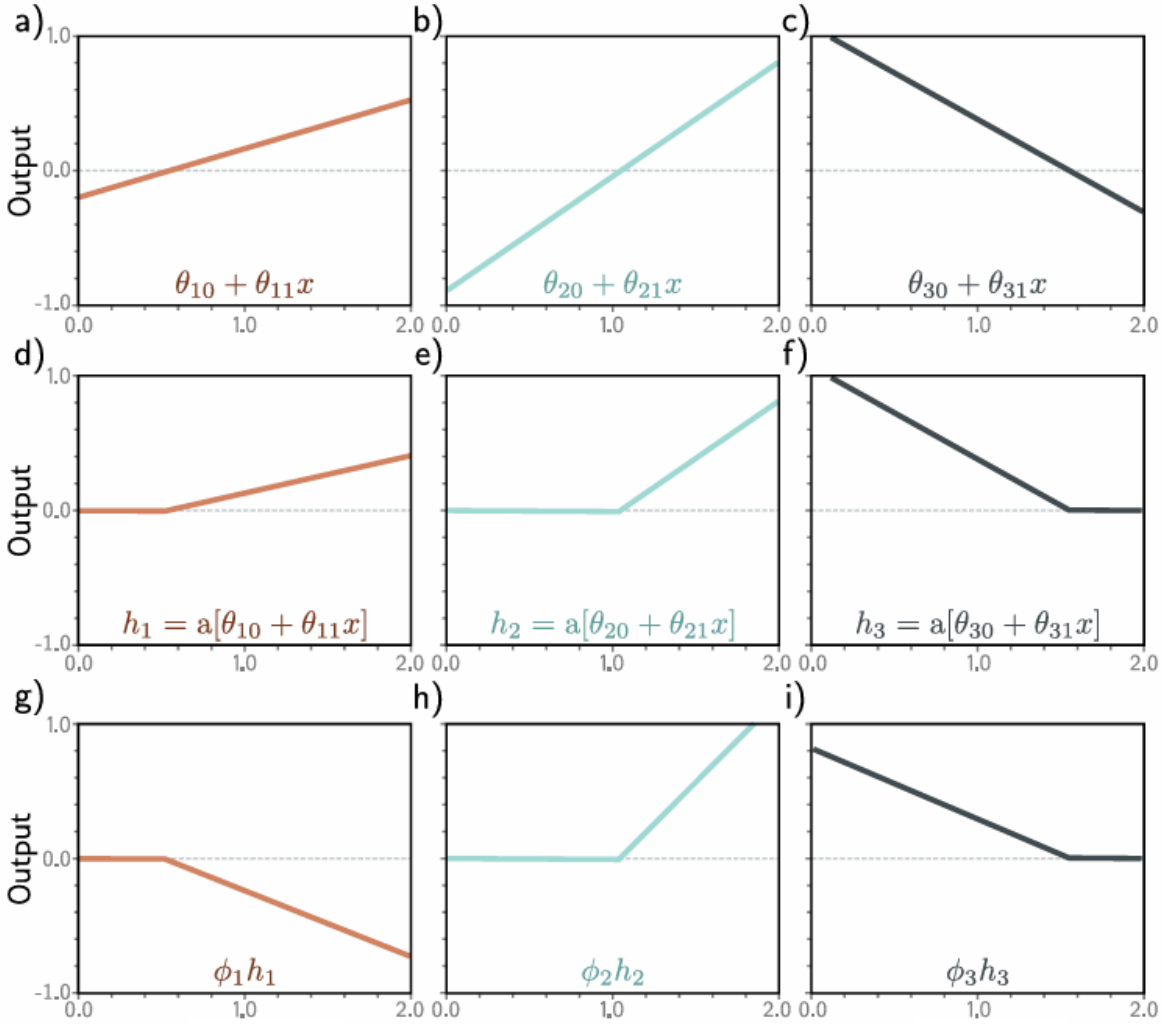
linear transformation in the first layer + activation



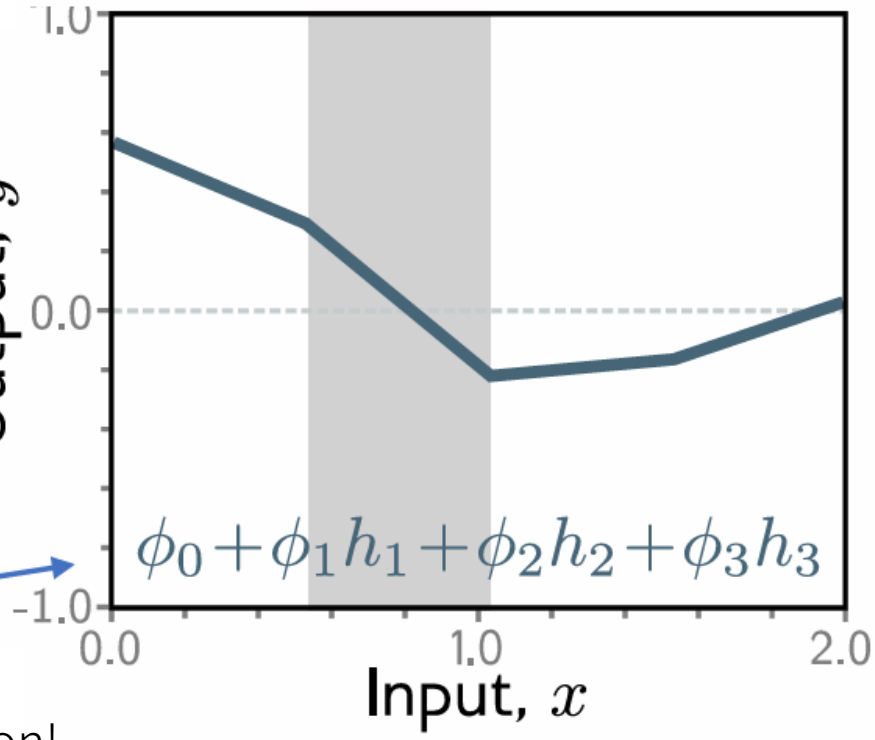
$$\mathrm{ReLU}(x) = \max(x, 0)$$

ReLU를 사용해서 예를 들어볼게요.

How does Neural Network approximate?

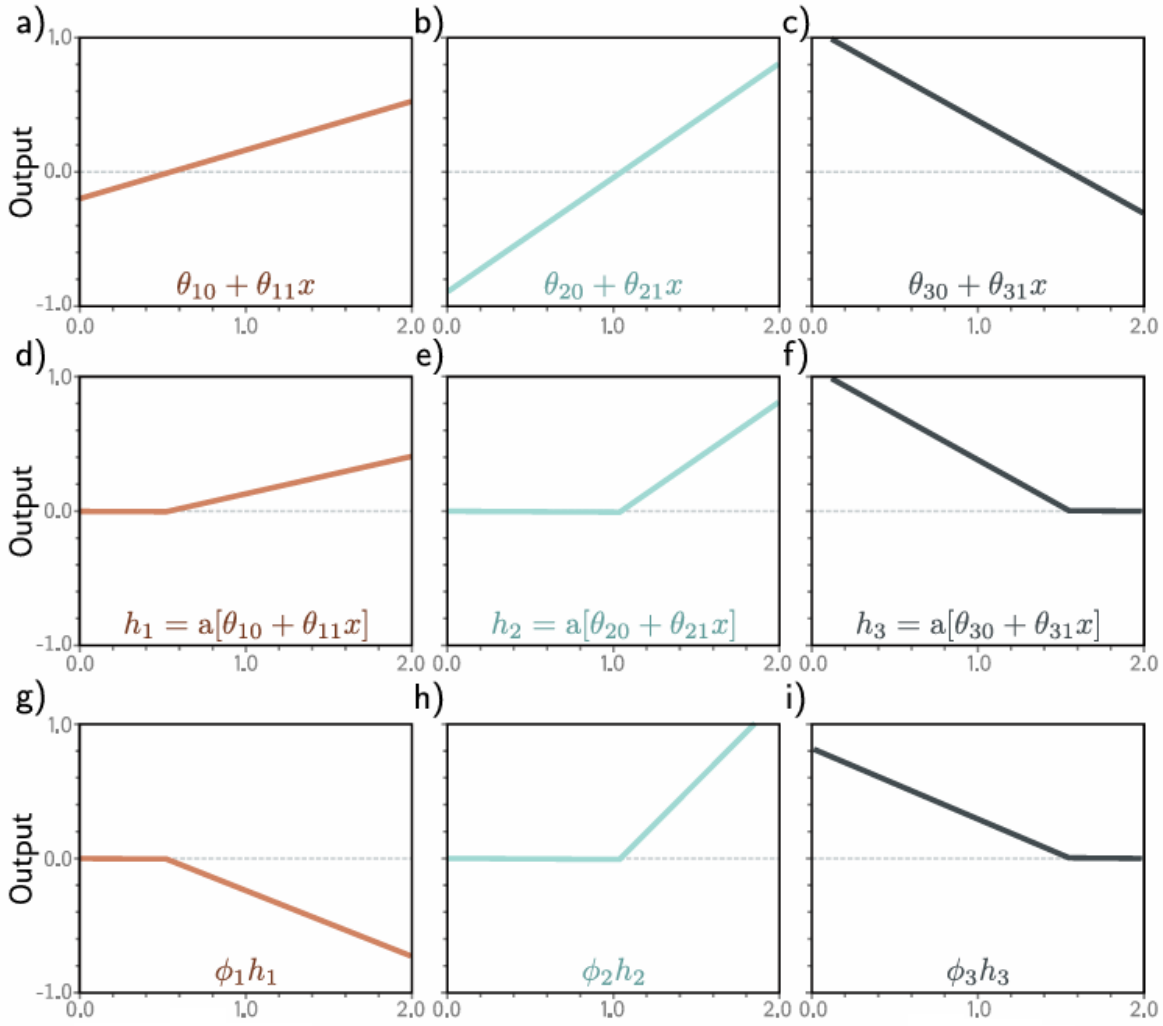


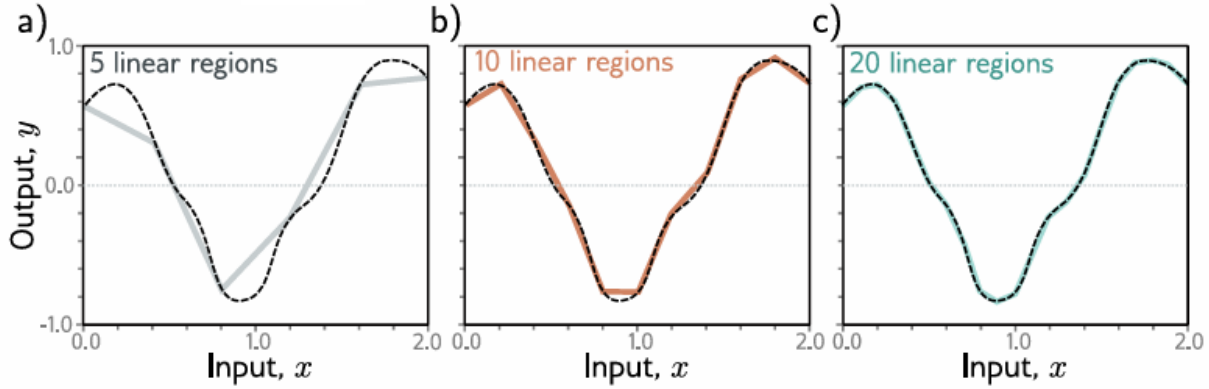
중첩이 비선형 함수를 만드는 것을 볼 수 있습니다



linear combination!

How does Neural Network approximate?



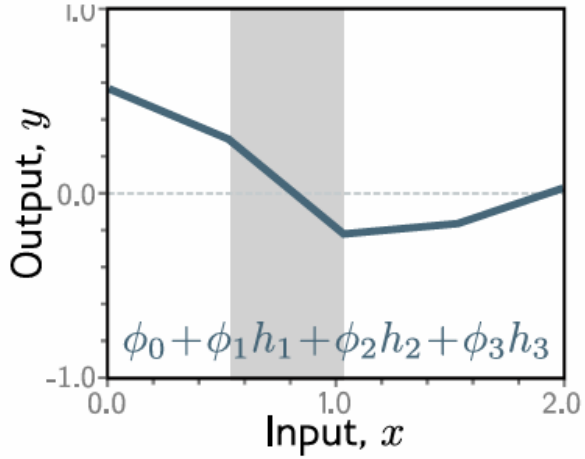


hidden variables의 수가 증가할수록

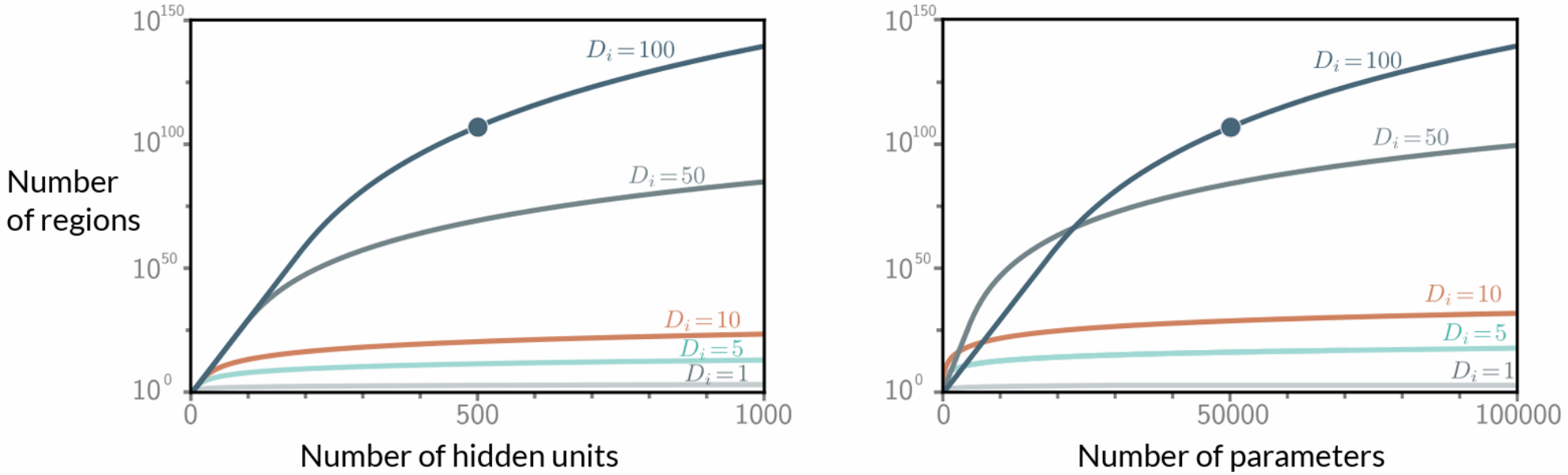
접히는 영역도 증가합니다

linear

combination!



Parameter size implies approximate power

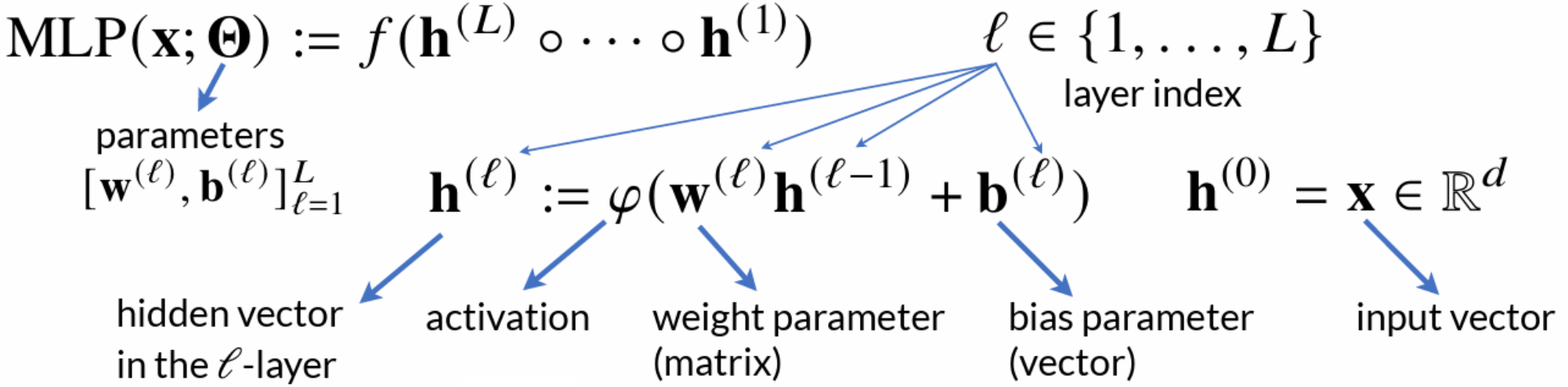


increases # of hidden units → increases # of parameters → increases # of regions

Neural networks의 power는 combinatorial folding에 있으며
이를 통해 고전 머신러닝이 겪던 차원의 저주를 극복합니다

Multi-Layer Perceptron

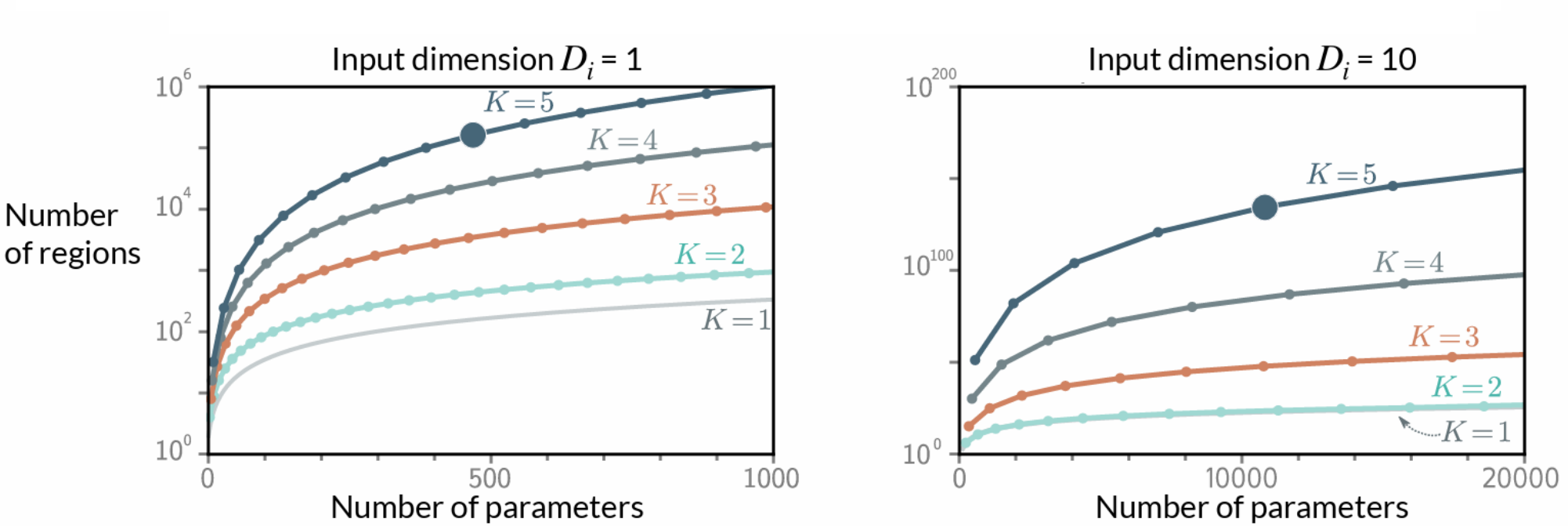
- 여러 퍼셉트론의 구성으로 MLP를 만들 수 있다.



- MLP는 파라미터가 몇 개일까?

$$|\Theta| \approx \sum_{\ell=1}^{L} \dim(\mathbf{h}^{(\ell)}) \cdot [\dim(\mathbf{h}^{(\ell-1)}) + 1]$$

Parameter size implies approximate power



increases # of layers (K) → increases # of regions

Neural networks가 깊을수록 더 많은 expressive power를 갖습니다

Theoretical Power of Deep Neural Networks?

- Universal Approximation Theorem (Hornik & Cybenko, 1989)
- Kolmogorov-Arnold Representation Theorem (Hilbert's 13th problem)
- For Deep Networks: Lu et al. (NeurIPS 2017) and Kidger & Lyons (2020)

Universal approximation theorem (Uniform non-affine activation, arbitrary depth, constrained width). Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^d$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be any non-affine continuous function which is continuously differentiable at at least one point, with nonzero derivative at that point. Let $\mathcal{N}^{\sigma}_{d,D:d+D+2}$ denote the space of feed-forward neural networks with d input neurons, D output neurons, and an arbitrary number of hidden layers each with $d + D + 2$ neurons, such that every hidden neuron has activation function σ and every output neuron has the identity as its activation function, with input layer ϕ and output layer ρ. Then given any $\varepsilon > 0$ and any $f \in C(\mathcal{X}, \mathbb{R}^D)$, there exists $\hat{f} \in \mathcal{N}^{\sigma}_{d,D:d+D+2}$ such that

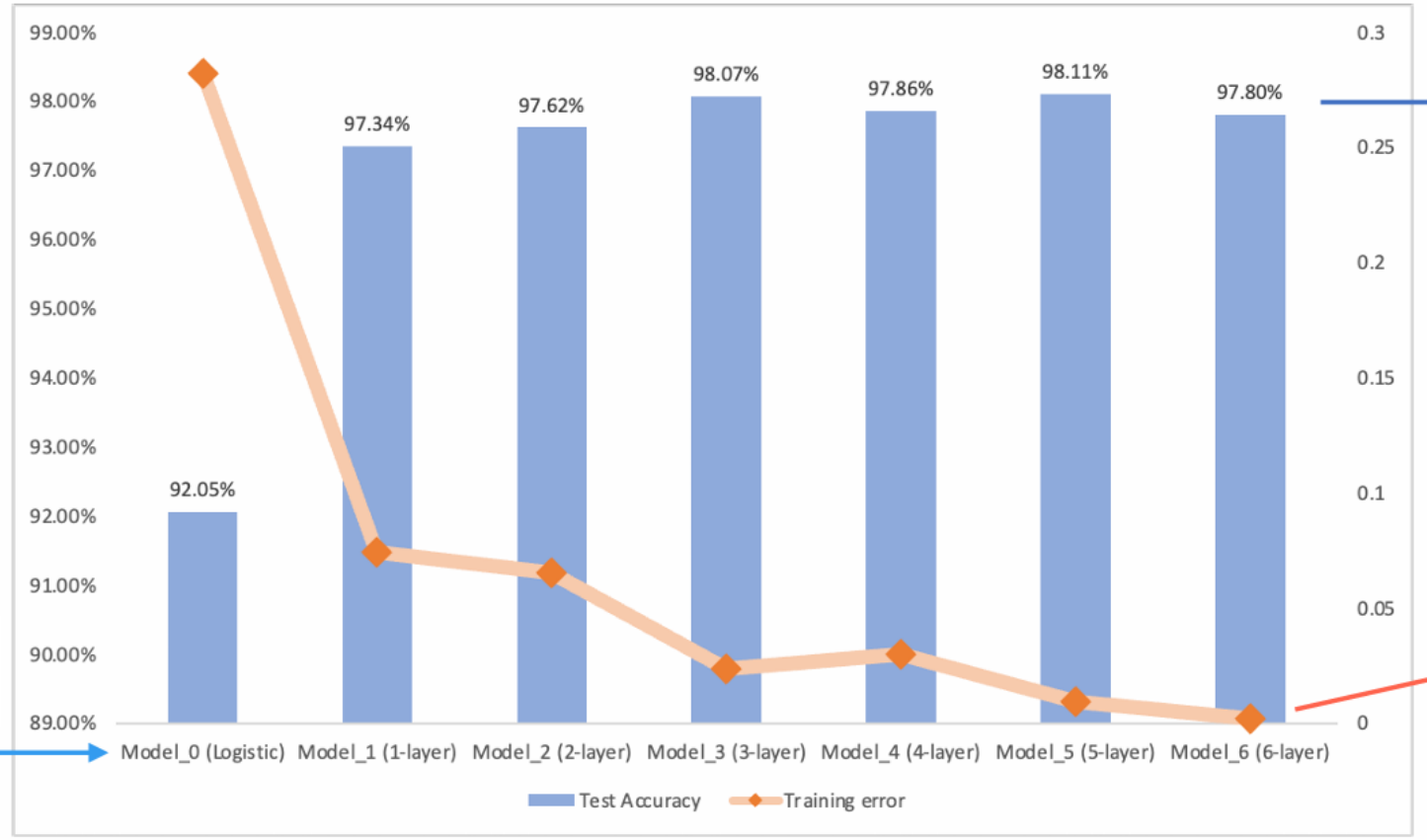
$$\sup_{x \in \mathcal{X}} \left\| \hat{f}(x) - f(x) \right\| < \varepsilon.$$

In other words, $\mathcal{N}$ is dense in $C(\mathcal{X}; \mathbb{R}^D)$ with respect to the topology of uniform convergence.

이 논문들은 deep neural networks에 대한 expressive power를 설명합니다

Universal Approximation with Deep Narrow Networks, Lu et al., NeurIPS 2017
The Expressive Power of Neural Networks: A View from the Width, Kidger & Lyons, ICLR 2020

More expressive power implies generalization?



layer의 수를 늘려도

test accuracy는 증가하지 않습니다

training error는 감소하지만

Universal Approximation Theorem does not imply that Deeper Networks generalize better

expressive power와 generalization은 다른 이야기입니다

Fundamental Problem of ML

머신러닝과 통계학에서 중요한 가정입니다

$$\mathbb{P}_{\text{train}} \approx \mathbb{P}_{\text{test}}$$

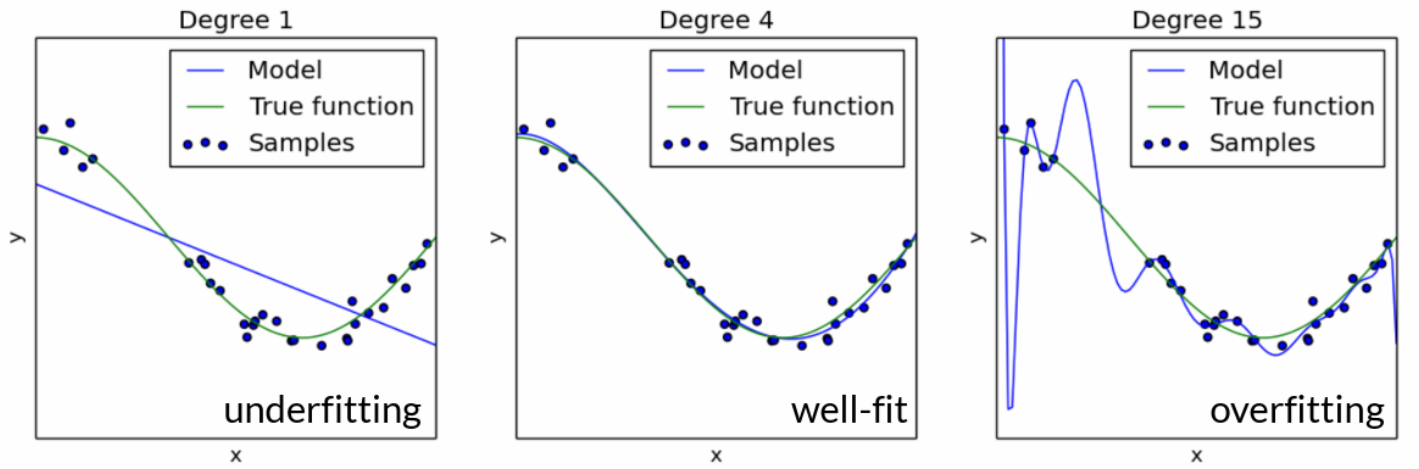
optimization

$$\min_{\mathbf{w}} \mathcal{L}(\mathcal{M}(\mathbf{w}), \mathcal{D}_{\text{train}})$$

capacity

$$\min_{\mathbf{w}} \mathcal{L}(\mathcal{M}(\mathbf{w}), \mathcal{D}_{\text{test}})$$

- Overfitting vs. Underfitting
  - Training error를 작게 만듭니다.
  - Training error와 test error의 차이를 작게 만듭니다.



Bias-Variance Tradeoff

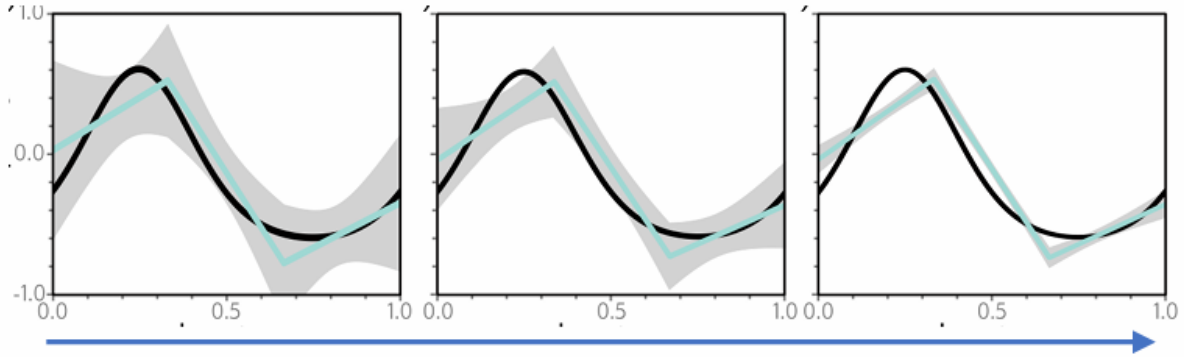
$$\text{mean}(\mathbf{x}) = \mathbb{E}_{y \sim \mathbb{P}(y|\mathbf{x})}[y] \qquad y \sim \mathbb{P}(y|\mathbf{x}) \qquad \mathcal{D} \sim \mathbb{P}(\mathbf{x}, y)$$ Data distribution

$$\mathbb{E}_{\mathcal{D} \sim \mathbb{P}(\mathbf{x},y)} \mathbb{E}_{y \sim \mathbb{P}(y|\mathbf{x})} \| f_{\theta(\mathcal{D})} - y \|_2^2 \qquad f_\Theta(\mathbf{x}) = \mathbb{E}_{\mathcal{D} \sim \mathbb{P}(\mathbf{x},y)}[f_{\theta(\mathcal{D})}(\mathbf{x})]$$ Expectation of model prediction

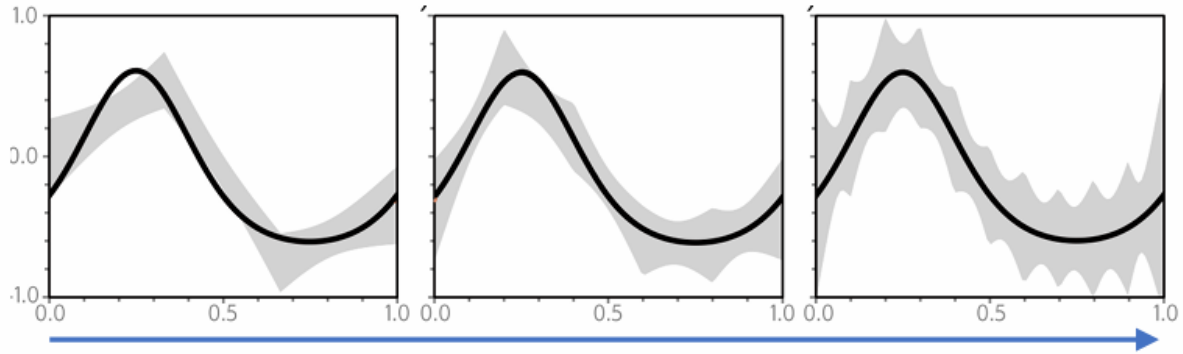
$$= \mathbb{E}_{\mathcal{D} \sim \mathbb{P}(\mathbf{x},y)} \| f_{\theta(\mathcal{D})}(\mathbf{x}) - f_\Theta(\mathbf{x}) \|_2^2 + \mathbb{E}_{\mathcal{D} \sim \mathbb{P}(\mathbf{x},y)} \| f_\Theta(\mathbf{x}) - \text{mean}(\mathbf{x}) \|_2^2 + \text{Noise}$$

Variance

Bias



more training data reduces *variance* but *bias* remains



more expressive power reduces *bias* but *variance* increases

For better generalization,
use larger models to reduce bias and
collect more data to reduce variance

Regularization

- 명시적 정칙화 (Explicit regularization)
  - weight decay (ridge regularization or L2 regularization)
  - dropout
  - data augmentation
  - early stopping
  - ensemble
- 암묵적 정칙화 (Implicit regularization)
  - Stochatstic Gradient Descent (SGD)

Weight decay

- Overfitting을 방지하는 방법 중 가장 대표적인 방법이다.
- 모델이 복잡할수록 패널티를 주어 지나치게 복잡한 모델이 되는 것을 막는다.

$$\mathcal{L}(\boldsymbol{\Theta}, \mathcal{D}_{\text{train}}) + \lambda \|\boldsymbol{\Theta}\|_2^2$$

weight decay term

$\Theta = \{\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(L)}\}$ model parameters

weight decay coefficient

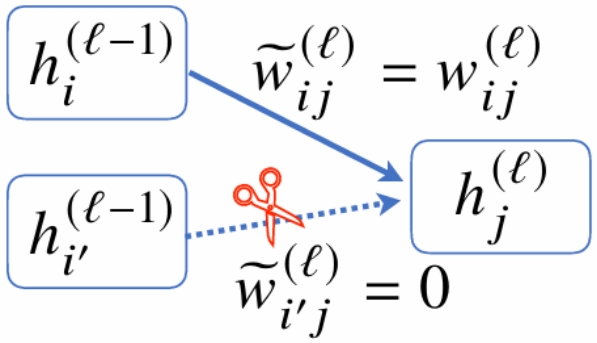
$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} (\mathcal{L} + \lambda \|\mathbf{w}\|_2^2) = (1 - \eta\lambda)\mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}$$

- 구현하기 굉장히 쉽다.

```
optimizer = torch.optim.SGD(model.parameters(),
                            lr=0.03,
                            weight_decay=0.1)
```

Dropout

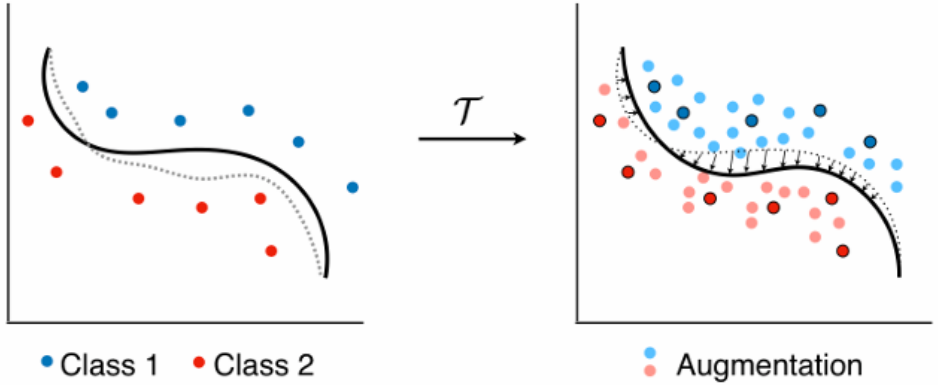
- Dropout은 간단하면서도 굉장히 강력한 방법으로, hidden variables 사이의 connection을 끊는 것이다.
  - 학습 중 가중치를 랜덤하게 0으로 만든다.
  - 예측 시 학습된 파라미터를 복원한다.
- 학습 결과가 확률적이게 되고 수렴에 방해가 되기도 한다.
- 사람의 좌뇌 우뇌 훈련과 비슷하다.

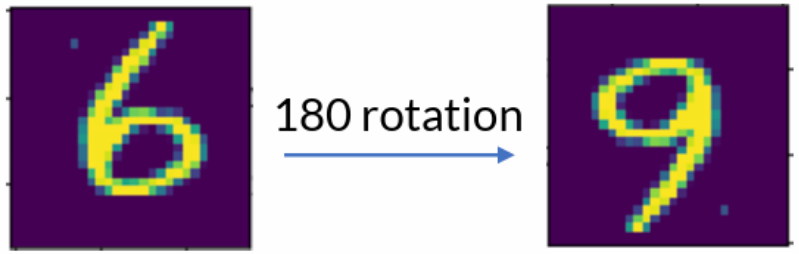


$$\tilde{w}_{ij}^{(\ell)} = \begin{cases} 0 & \text{with probability } p \\ w_{ij}^{(\ell)} & \text{with probability } 1-p \end{cases}$$

Data Augmentation

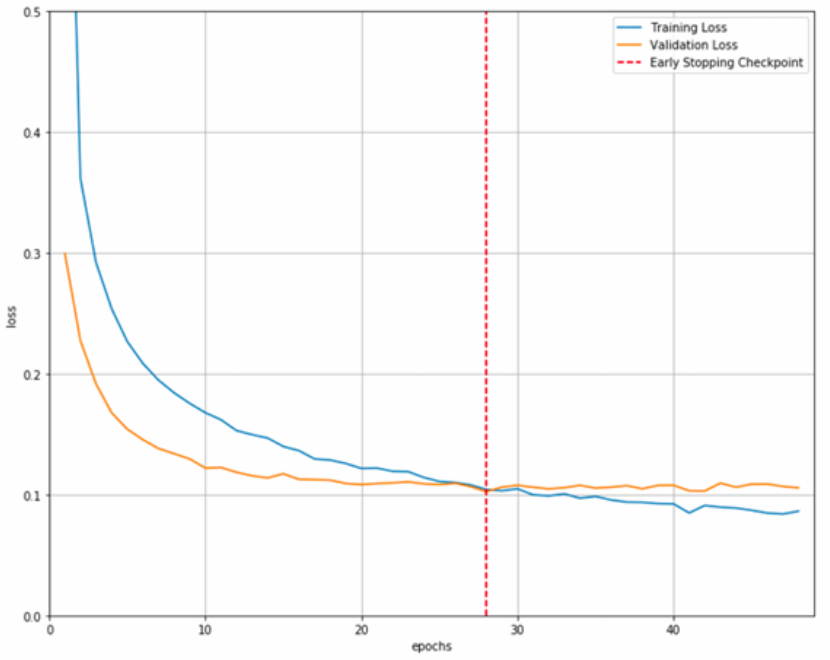
- 충분한 데이터가 없을 때, 데이터를 증강시켜 모델 학습에 사용할 수 있다.
  - 경험적으로, Data Augmentation은 데이터 문제 해결에 있어 가장 좋은 전략이다.
  - Pros
    - Neural networks에 잘 맞는 방법이다.
    - 모델을 변경하지 않아도 된다.
    - 따라서 모델을 튜닝하는 것보다 더 쉽다.
  - Cons
    - Domain knowledge가 필요하다.
    - 가끔 더 많은 학습을 요구하기도 한다.





Early Stopping

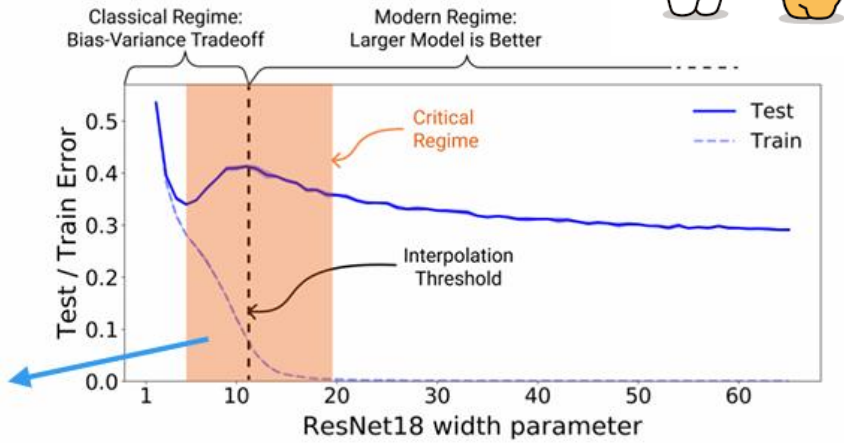
- 모델이 overfitting되기 시작하면 학습을 중단하는 방법이다.
  - cross validation이 반드시 필요하다. (절대 test set을 써서는 안된다!)
  - Pros
    - 간단하면서도 강력하다.
    - 모델 및 테스크에 agnostic하다.
  - Cons
    - 휴리스틱이 필요하다.
    - double descent와 모순된다.

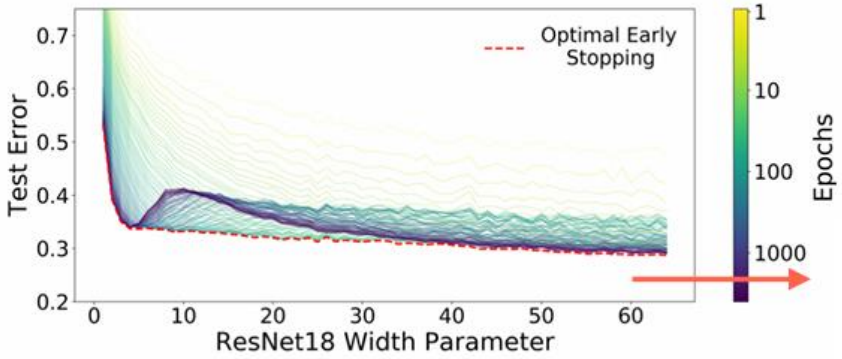


Deep Double Descent

고전적인 믿음도 경험적 증거에 의해 깨지고 있습니다

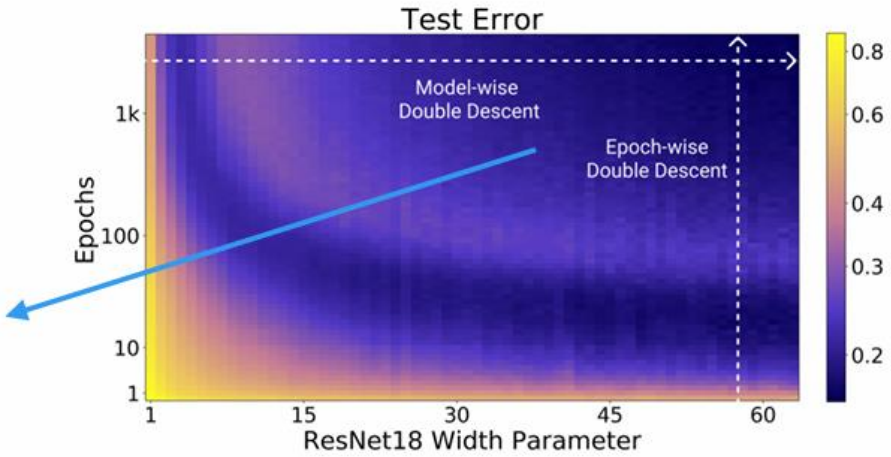
현대 딥러닝은 더 많은 파라미터가 있을수록 더 좋은 성능을 냅니다

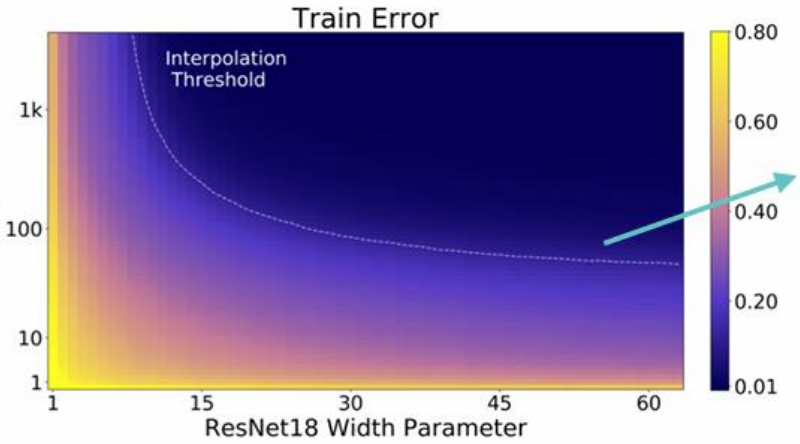




Early stopping은 상대적으로 파라미터가 적을 때 유효합니다

만약 모델이 충분히 크다면 오래 학습하는 것이 효과적입니다





training error가 0이 되도록 네트워크를 학습하는 것이 모델의 퍼포먼스를 향상시킵니다

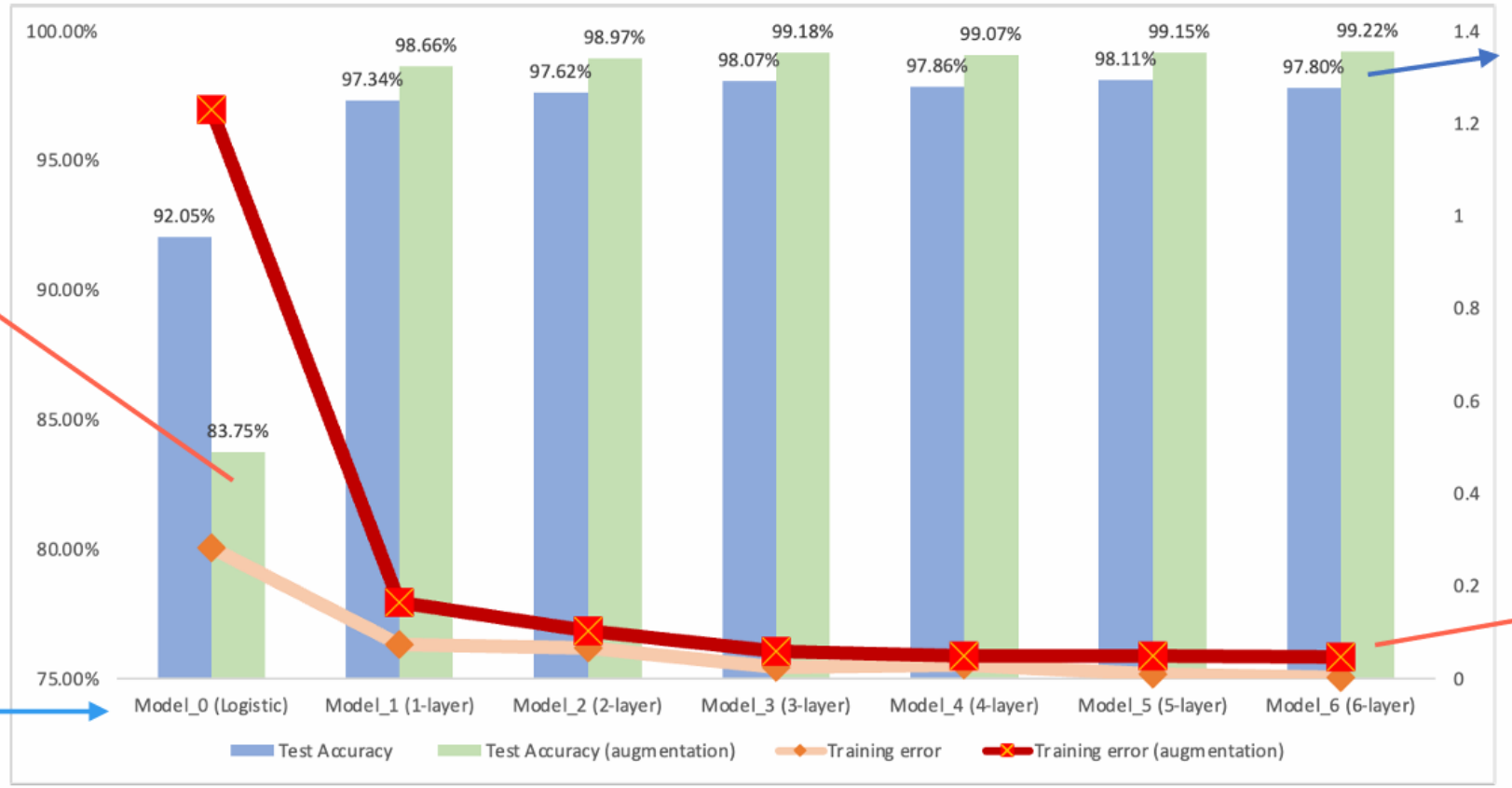
Deep Double Descent: Where Bigger Models and More Data Hurt, Nakkiran et al., ICLR 2020

More expressive power implies generalization!

with data augmentation + *longer* training

test accuracy도
높일 수 있습니다

작은 모델은 안 좋은
성능을 내지만



training error는
감소하며 동시에

layer의 수를 늘리면

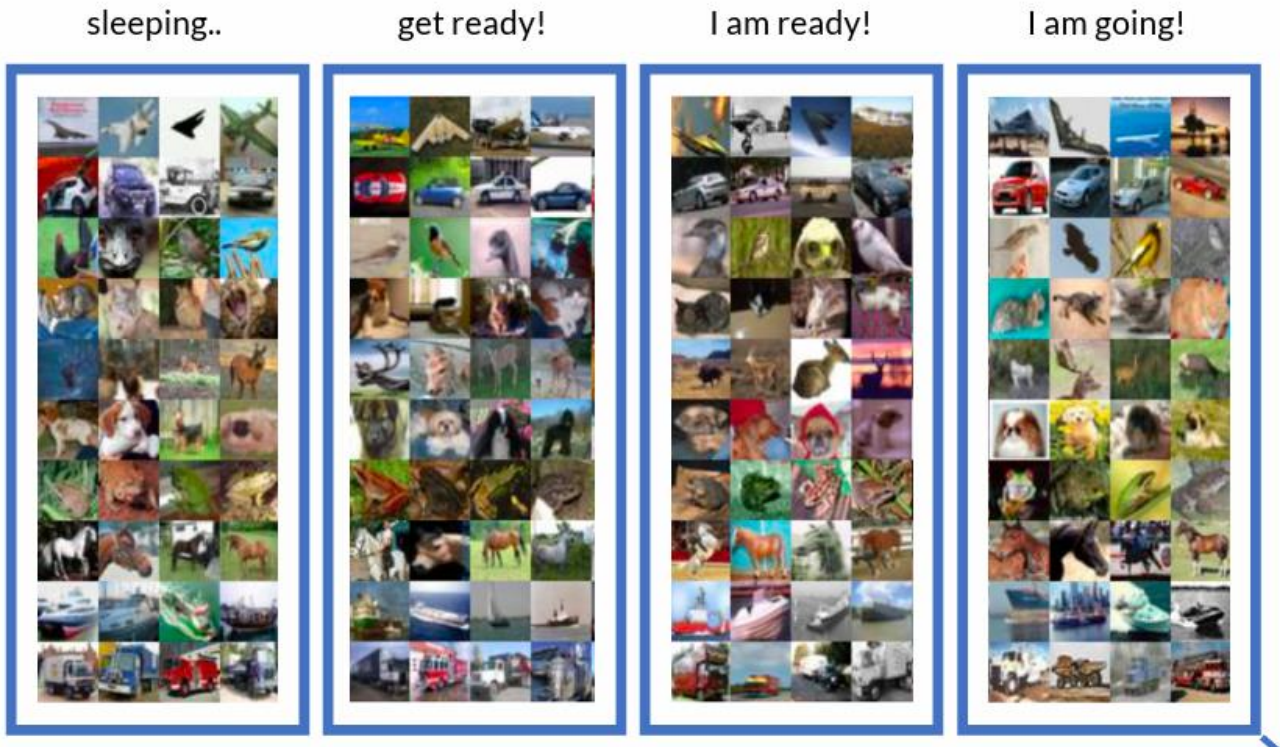
Summary

- 일반화(Generalization)은 머신러닝 알고리즘의 핵심이다.
  - 머신러닝 모델은 bias와 variance 간의 tradeoff 관계에 놓여있다.
  - 일반화를 위해서는 모델 사이즈를 키워 bias를 줄이고 더 많은 데이터로 학습해 variance를 줄여라.
- Universal Approximation Theorem은 딥러닝의 존재성에 대한 이론으로 일반화를 설명하는 것은 아니다.
  - Overfitting을 해결하기 위해 weight decay, dropout, data augmentation, early stopping 등이 자주 사용된다.
- 최근 연구에 따르면 large neural networks는 모델 파라미터와 학습 시간에 따른 double descent 현상을 지닌다.
  - 경험적으로, 네트워크가 깊을수록 얕은 네트워크보다 loss 함수를 smoothing하는 것이 알려져 있다.
  - 최적화(Optimization)와 학습(training) 방법론이 매우 중요하다.

A Universal Law of Robustness via Isoperimetry, S. Bubeck & M. Selke, NeurIPS 2021

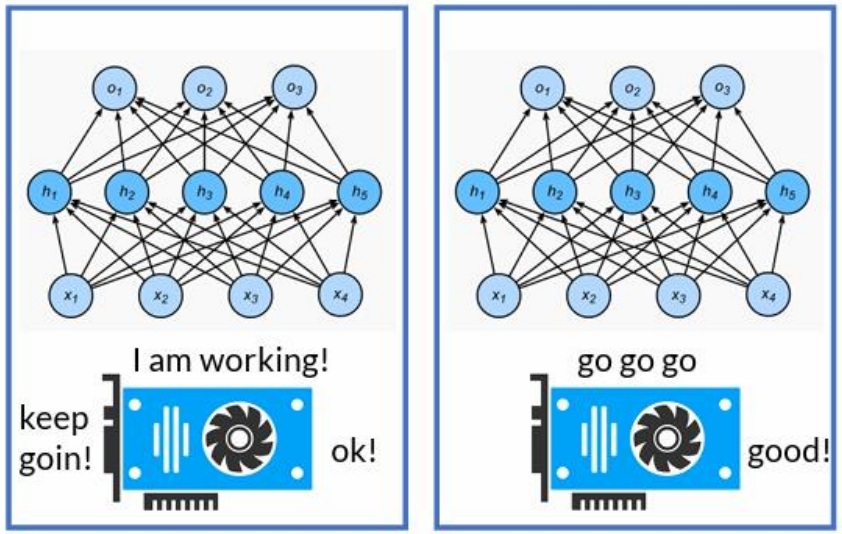
One more thing..

Deep Learning with Big Data



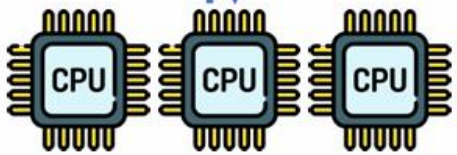
$$256 \times 256 \times 3 \times |\mathcal{B}| \leq 2^{18} \cdot |\mathcal{B}|$$

Image data



Training
- matrix calculus
- backprop
- parallel computing
- ...

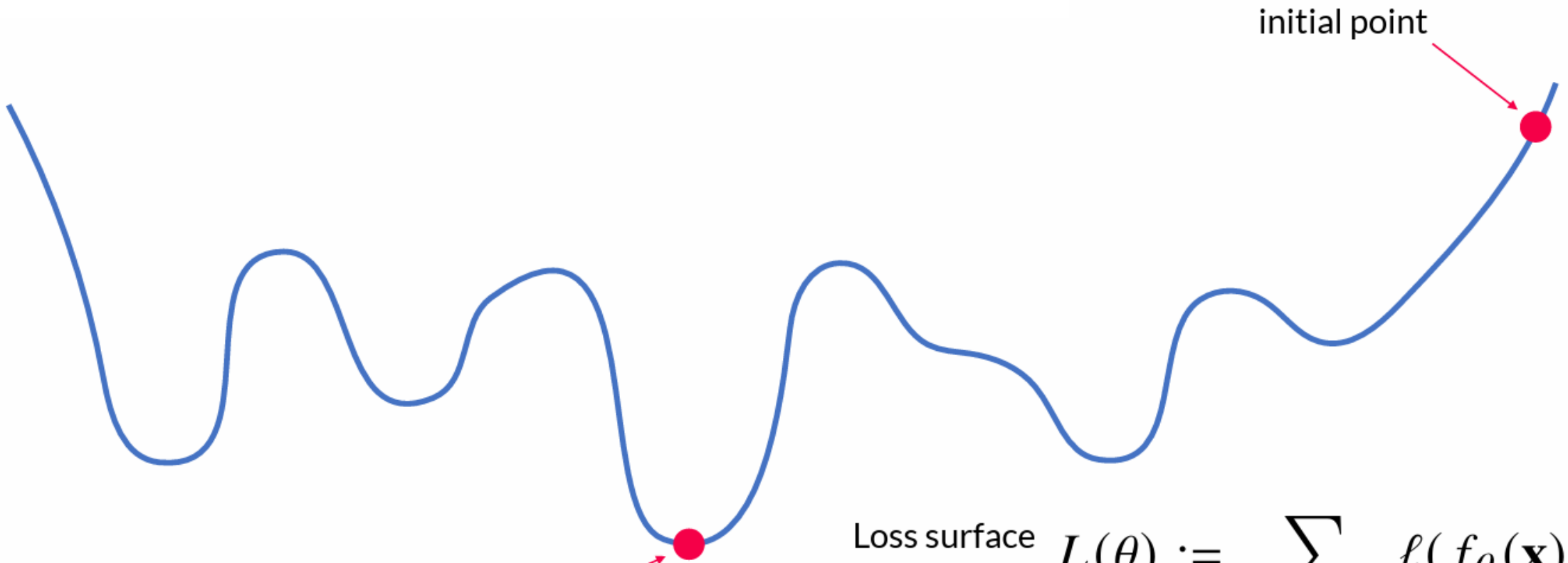
much easier than before



Working
- read & load data
- preprocessing
- aggregate
- ...

전체 데이터를 minibatch로 나눠서 순차적으로
학습하는 전략이 현대 딥러닝의 최적화 기법입니다

Workhorse of Deep Learning



Loss surface for train data $L(\theta) := \sum_{(\mathbf{x},y)\in\mathcal{D}} \ell(f_\theta(\mathbf{x}), y)$

$(\mathbf{x},y)\in\mathcal{D}$: train data

$\ell(f_\theta(\mathbf{x}), y)$: Loss at $(\mathbf{x}, y)$

.

Workhorse of Deep Learning

Full batch에 대한 gradient descent는
local optimum으로 수렴합니다

Workhorse of Deep Learning

$$L(\theta) := \sum_{(\mathbf{x},y)\in\mathcal{D}} \ell(f_\theta(\mathbf{x}), y)$$

Loss surface — train data — Loss at $(\mathbf{x}, y)$

GD Algorithm $$\theta_{t+1} \leftarrow \theta_t - \eta_t \nabla L(\theta_t)$$

Stochastic GD Algorithm $$\theta_{t+1} \leftarrow \theta_t - \eta_t \nabla \ell(f_{\theta_t}(\mathbf{x}), y)$$

Robbins & Monro condition $$\sum_t \eta_t = \infty \quad \sum_t \eta_t^2 < \infty$$

R-M condition은 SGD의 수렴을 결정합니다

-

Workhorse of Deep Learning

$$L(\theta) := \sum_{(\mathbf{x},y)\in\mathcal{D}} \ell(f_\theta(\mathbf{x}), y)$$

Loss surface — train data — Loss at $(\mathbf{x}, y)$

GD Algorithm $$\theta_{t+1} \leftarrow \theta_t - \eta_t \nabla L(\theta_t)$$

Stochastic GD Algorithm $$\theta_{t+1} \leftarrow \theta_t - \eta_t \nabla \ell(f_{\theta_t}(\mathbf{x}), y)$$

Robbins & Monro condition $$\sum_t \eta_t = \infty \quad \sum_t \eta_t^2 < \infty$$

Minibatch Stochastic GD Algorithm $$\theta_{t+1} \leftarrow \theta_t - \eta_t \sum_{(\mathbf{x},y)\in\mathcal{B}} \nabla \ell(f_{\theta_t}(\mathbf{x}), y)$$

$\mathcal{B}$ → minimatch $\mathcal{B} \subset \mathcal{D}$

Workhorse of Deep Learning

Full batch에 대한 gradient descent는
local optimum으로 수렴합니다

Workhorse of Deep Learning

Minibatch에 대한 random selection이
local optimum 탈출에 도움이 됩니다

Loss surface for *minibatch* $L(\theta|\mathcal{B}) := \sum_{(\mathbf{x},y)\in\mathcal{B}} \ell(f_\theta(\mathbf{x}), y)$

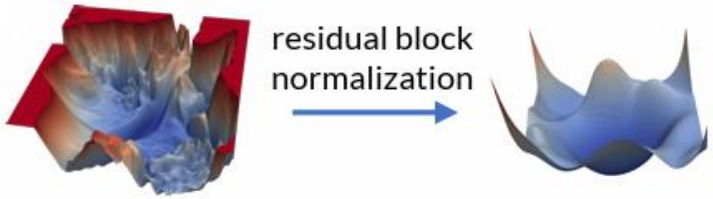
Loss at $(\mathbf{x}, y)$

minibatch train data

Workhorse of Deep Learning

ResNet에서 배울 residual connection과

normalization이 loss surface를 flatten시킵니다



initial point

flat minima

$\nabla_\theta L(\theta|\mathcal{B})$

Gradient w.r.t. θ

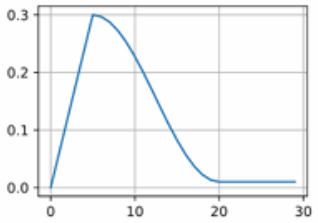
Loss surface for *minibatch* $L(\theta|\mathcal{B}) := \sum_{(\mathbf{x},y)\in\mathcal{B}} \ell(f_\theta(\mathbf{x}), y)$

Loss at $(\mathbf{x}, y)$

minibatch train data

Workhorse of Deep Learning

Warmup 같은 learning rate scheduler는
local minima 탈출에 도움이 됩니다



initial point

initial point

flat minima

$\nabla_\theta L(\theta|\mathcal{B})$

$\nabla_\theta L(\theta|\mathcal{B})$
Gradient w.r.t. θ

Loss surface for *minibatch* $L(\theta|\mathcal{B}) := \sum_{(\mathbf{x},y)\in\mathcal{B}} \ell(f_\theta(\mathbf{x}), y)$

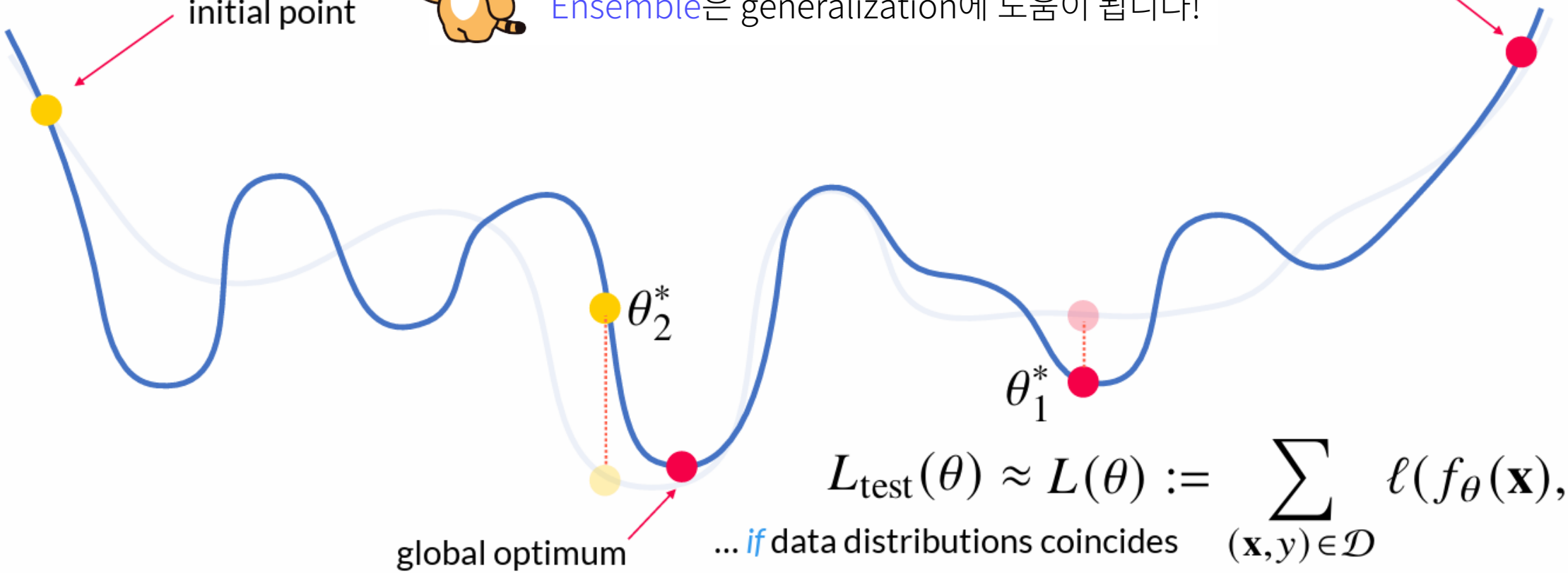
Loss at $(\mathbf{x}, y)$

minibatch train data

Workhorse of Deep Learning

여러 시작점에서의 수렴 결과를 반영하는

Ensemble은 generalization에 도움이 됩니다!



$$L_{\text{test}}(\theta) \approx L(\theta) := \sum_{(\mathbf{x},y)\in\mathcal{D}} \ell(f_\theta(\mathbf{x}), y)$$

... *if* data distributions coincides

$$\mathbb{P}_{\text{test}} = \mathbb{P}_{\text{train}}$$

Does SGD Algorithm work theoretically?

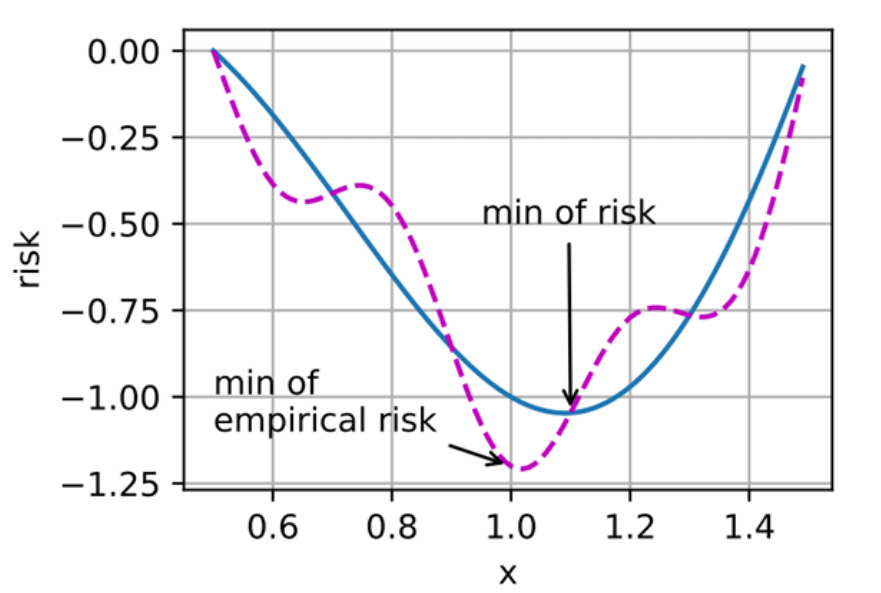
- 다음 질문에 답을 해보자.
  - SGD는 stationary point로 수렴하는가?
  - SGD는 saddle point 같은 critical regions를 피할 수 있는가?
  - SGD가 local optimum에 수렴하는 속도는?
  - SGD는 언제 global optimum에 수렴할 수 있을까?

Does SGD Algorithm work theoretically?

- 다음 질문에 답을 해보자
  - SGD는 stationary point로 수렴하는가?
    - 네!
  - SGD는 saddle point 같은 critical regions를 피할 수 있는가?
    - 네!
  - SGD가 local optimum에 수렴하는 속도는?
    - $\mathcal{O}(n^{-p})$
  - SGD는 언제 global optimum에 수렴할 수 있을까?
    - Strongly convex / Polyak-Łojasiewicz condition

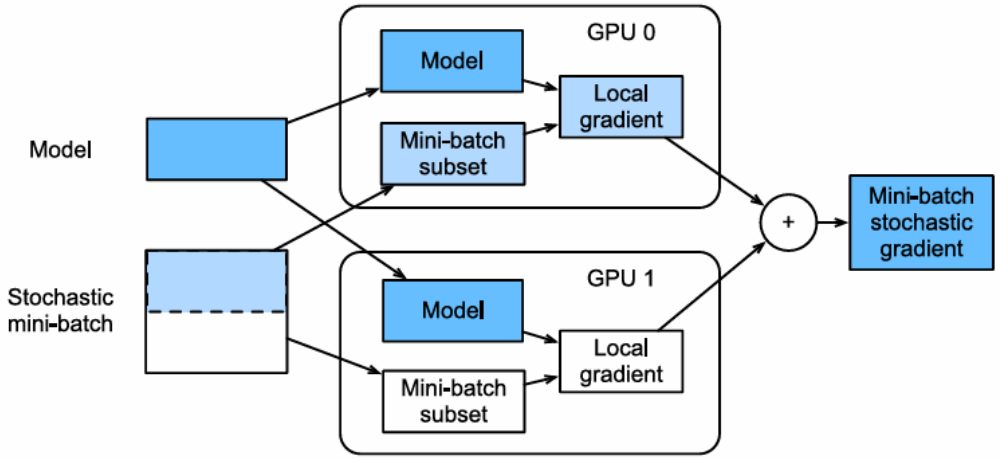
Optimization ≠ Deep Learning

- 딥러닝에서 발생하는 거의 대부분의 optimization 문제는 non-convex이다.
- 딥러닝은 결국 최적화 문제를 푸는 것이라고 주장하는 사람들이 있지만 이는 사실이 아니다.
  - 최적화의 목표는 training error를 줄이는 것이다.
  - 딥러닝의 목표는 generalization error를 줄이는 것이다.

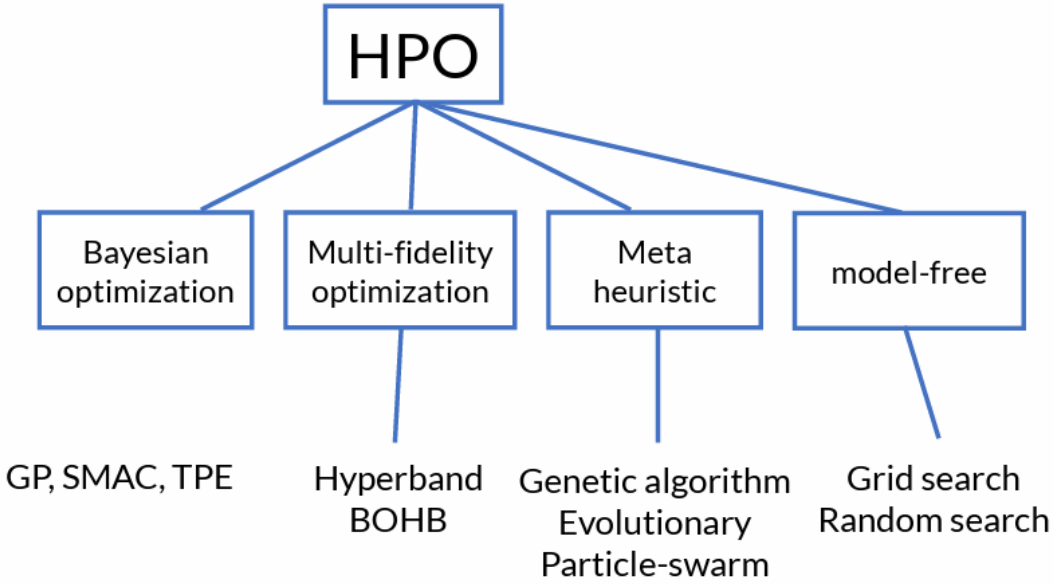


$$L_{\text{train}} \neq L_{\text{test}}$$

Modern Optimization Principles



Large-scale Stochastic Optimization

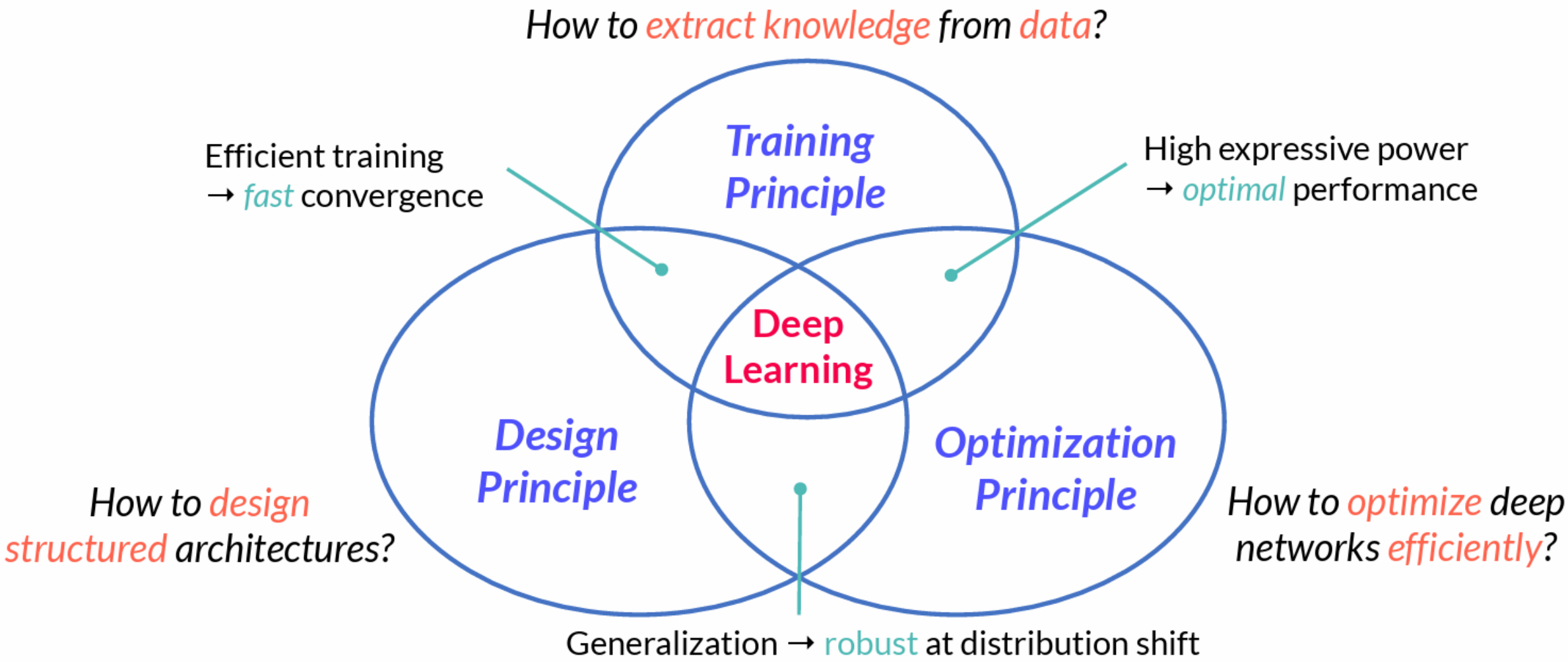


Model Selection & HPO

What will you learn from next?

- 딥러닝의 원리 (Principles of Deep Learning)
  - Design principles: invariance, neural architectures, complexity
  - Optimization principles: stochastic optimization, model selection
  - Training principles: self-supervised learning, foundation models (이번 AI week에서 다루지는 않습니다)

Principles of Deep Learning



Quiz #1

- MNIST 분류 문제에서 test accuracy를 계산하는 코드를 구현하고 checkpoint 생성
  - 필수조건
    - 모델 학습 코드(train_model) 내부에 구현
    - Checkpoint에 model, optimizer, test accuracy 반드시 포함

- Github repo에 코드를 올려두기

Quiz #2

- 모듈화한 학습코드를 활용하여 CIFAR-10 데이터 학습
  - 필수조건
    - CIFAR 데이터로더는 torchvision 내부 함수 활용
    - 5회 반복 측정한 평균 test accuracy 측정
  - 도전
    - MNIST에서 학습된 모델을 CIFAR 데이터에서 fine-tuning하기

- Github repo에 코드를 올려두기

KUGODS